Reservoir Computing with high non-linear separation and long-term memory for time-series data analysis

John B Butcher

Submitted in partial fulfilment of the requirements of the degree of

Doctor of Philosophy

March 2012

Keele University

Abstract

Left unchecked the degradation of reinforced concrete can result in the weakening of a structure and lead to both hazardous and costly problems throughout the built environment. In some cases failure to recognise the problem and apply appropriate remedies has already resulted in fatalities. The problem increases with the age of any structures and consequently has become more pressing throughout the latter half of the 20^{th} century. It is therefore of paramount importance to assess and repair these structures using an accurate and cost-effective approach. ElectroMagnetic Anomaly Detection (EMAD) is one such approach where currently analysis is performed visually, which is undesirable. A relatively new Recurrent Artificial Neural Network (RANN) approach which overcomes problems which have prohibited the widespread use of RANNs, Reservoir Computing (RC), is investigated here.

This research aimed to automate the detection of defects within reinforced concrete using RC while gaining further insights into fundamental properties of an RC architecture when applied to real-world time-series datasets. As a product of these studies a novel RC architecture, Reservoir with Random Static Projections (R^2SP), has been developed. R^2SP helps to address what this research shows to be an antagonistic trade-off between a standard RC architecture's ability to transform its input data onto a highly non-linear state space whilst at the same time possessing a short-term memory of its

previous inputs.

The R^2SP architecture provided a significant improvement in performance for each dataset investigated when compared to a standard RC approach as a result of overcoming the aforementioned trade-off. The implementation of an R^2SP architecture is now planned to be incorporated on a new version of the EMAD data collection apparatus to give fast or near to real-time information about areas of potential problems in real-world concrete structures.

Keywords: Reservoir Computing, Reservoir with random static projections, Electromagnetic anomaly detection, Reinforced Concrete, Non-linearity, Short-term memory, Time-series data

Acknowledgements

This research was funded through an industrial CASE studentship award provided by the UK Engineering and Physical Sciences Research Council (EPSRC) and SciSite Ltd.

I would like to say a huge thanks to all the many people whom without this research would have been possible. A quick glance over the next two pages will indicate how many people have been involved in this project and helped over its duration, all of whom I have tried to mention. Inevitably, however, there will be someone I have forgotten, to whom I apologise and thank in advance. Firstly, to my supervisors Prof. Peter Haycock and Dr. Charles Day: to Peter for offering me the chance to take on a PhD and for his ever helpful comments, encouragement and wisdom he always provided; and to Charles, for without his passion for computational intelligence and particularly neural networks I experienced during my undergraduate studies, I would probably have never become interested in this stimulating and exciting field. I only hope one day I can instil my enthusiasm of research to inspire students to take an interest also. For this reason I can forgive you for being a Baggies fan. I would also like to thank past and present members of the research group I was lucky enough to meet and spend some time with during my studies including, Dr. Matthew Hocking, Dr. Sarah Sherratt, Mike Lion, Dr. Lawrence North and Dr. Jim Austin: in particular, Matthew and Jim for their

kind help in fixing the probe and for Mike, Sarah and Lawrence for your help, and to you all for making it such a great group in which to work. I would also like to say thanks to the guys in the office I shared during my time at Keele: Clive Jefferies, Ryad Soobhany, Dr. Siffatulah Khan, James Rooney, Louis Major, Liz Aston, Dr. Kappila Ponnamperuma and Dr. Phil Woodall: you all made working in the attic memorable and always fun which helped me through days when computers did not want to know. An additional thanks must be made to James for the help and patience he offered while I was trying to set up my own grid to run reservoir simulations. I also have to say a big thanks to all the support staff in the Computer Science and Chemistry departments, in particular Ian Marr for suffering my constant requests of additional PCs and screwdriver loans, and Ralph Pattison, the most go out and get it person I will probably ever meet - without you I would have struggled immensely. A big thanks goes to Alan Dudley for his help with the greenhouse experiment, his constant banter and his ability to make me laugh on a regular basis. I must also thank Dr. John Broomfield, for his invaluable advice regarding the mesh experiment that formed the basis for my data collection, for his permission to use images from his excellent book and for being a nice guy and always finding time to reply to my e-mails. I cannot finish my acknowledgements without thanking all of the guys from the ELIS department in Ghent where I spent some time during my research; not only did you always give me indispensable advice and knowledge, but you are also a very cool bunch of guys to work with who always listened to my ideas, gave good feedback and always made my stay in Ghent a pleasurable one, so thank you. I'm especially thankful to Dr. David Verstraeten for his patience while I got to learn how to use, and then break the grid, and all of your invaluable suggestions over the time we've got to know each other and your invaluable replies to the many emails I sent, and to Prof. Benjamin Schrauwen,

who initially invited me to give a talk in their department and soon found that he could not get rid of me, and who also provided the inspiration behind much of this research work for which I am hugely indebted. I also want to say a huge thanks to Pieter Buteneers, Michiel Hermans and Tim Waegemans for providing me with a desk in their office and providing such a stimulating and enjoyable working environment. I must thank Michiel again for putting up with me most nights whilst I was in Belgium and showing me the wonderful sights, beers, food and people that Ghent has to offer. I must also thank Bob Eyre for reading my thesis and spotting the many typos and my use of poor grammar which I insisted on putting in here. I must say a huge thank you to my family and friends who have helped keep things in perspective and when needed offered a welcome distraction. In particular my Mum and Dad, Chris (thanks for the year loan of your transformer too!), Laura and Claire for their suffering and patience over the years. And of course, to Hanny, for listening to my many moans and groans about theses and reservoirs; somehow you always find some way to make me smile and appreciate life, without you I would not be half the person I am today. Finally, this is for you Nan - you were an inspiration to us all.

Contents

List of Figures

List of Tables

List of Acronyms

ANN	Artificial Neural Network
ASE	Average State Entropy
BPDC	Backprogation Decorrelation
BPTT	Backpropagation through time
CI	Computational Intelligence
DT	Destructive Testing
DTDNN	Distributed Time-Delay Neural Network
EMAD	Electromagnetic Anomaly Detection
EN	Elman network
EN-BPTT	Elman network trained using Backpropagation Through Time
EN-RTRL	Elman network trained using Real-Time Recurrent Learning
ESN	Echo State Network
ELM	Extreme Learning Machine
FFT	Fast-Fourier Transform
HMM	Hidden Markov Model
LLE	Local Lyapunov Exponent

LMS	Least Mean Squares
LRN	Layer Recurrent Network
LSM	Liquid State Machine
LSTM	Long Short-Term Memory
MC	Memory Capacity
MFCC	Mel-Frequency Cepstral Coefficient
MLP ANN	Multi-Layer Perceptron ANN
MRSR	MultiResponse Sparse Regression
NARMA	Non-linear Auto Regressive Moving Average
NDT	Non Destructive Testing
NRMSE	Normalised Root Mean Squared Error
OSBP-ESN	One Step Backpropagation ESN
PCA	Principle Component Analysis
RANN	Recurrent Artificial Neural Network
RC	Reservoir Computing
Rebar	Steel reinforcing bar
R^2SP	Reservoir with Random Static Projections
RLS-ESN	Recursive Least Squares ESN
RTRL	Real-Time Recurrent Learning
SLR	Systematic Literature Review
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine
TD-ELM	Time-delay Extreme Learning Machine

TRM	Temporal Reservoir Machine
WER	Word Error Rate

List of symbols

$\mathbf{X}$	Matrix of reservoir activations
$\mathbf{x}$	Vector of a reservoir neuron's activation
$\mathbf{W}_{res}^{inp}$	Input-to-reservoir weight matrix
$\mathbf{W}_{res}^{res}$	Reservoir-to-reservoir weight matrix
$\mathbf{W}_{out}^{res}$	Reservoir-to-output weight matrix
$\mathbf{W}_{out}^{bias}$	Bias-to-output weight matrix
$\mathbf{W}_{out}^{inp}$	Input-to-output weight matrix
$\mathbf{W}_{res}^{bias}$	Bias-to-reservoir weight matrix
$\mathbf{W}_{res}^{out}$	Output-to-reservoir weight matrix
$\mathbf{u}$	Vector of input data
$\mathbf{y}_{tgt}$	Vector of target output data
$\hat{\mathbf{y}}$	Vector of actual network outputs
$\| \ldots \|_2$	Euclidean norm
N	Number of reservoir neurons
T	Total number of time steps of a dataset
ι	Input-to-reservoir scale

ρ	Maximal eigenvalue of $\mathbf{W}_{res}^{res}$ (spectral radius)
ρ_{eff}	Effective spectral radius
δ	Leak rate
δ_{ϕ}	Deviation from linearity
β	Bias-to-reservoir scalar
λ	Optimal regularisation parameter
$\widetilde{LLE}_{max}$	Estimated maximal local Lyapunov Exponent
$\mathbf{J}_f(X)$	Jacobian of matrix X
$\mathbf{X}_{stat_1}$	Activation matrix of static layer 1 neurons
$\mathbf{X}_{stat_2}$	Activation matrix of static layer 2 neurons
$\mathbf{W}_{stat_1}^{inp}$	Input-to-static layer 1 weight matrix
$\mathbf{W}_{stat_2}^{res}$	Reservoir-to-static layer 2 weight matrix

Chapter 1

Introduction

1.1 The widespread use of concrete

Concrete has been a part of our way of life for many years, with evidence suggesting that our use of concrete may stretch as far back as 4,000 years ago from the days of the ancient Egyptians [1], enabling us to build infrastructure to support our daily activities, open new avenues of technological advancement and improve our efficiency of tasks which we now take for granted but probably would not be possible without concrete. Its uses range from motorways, bridges, dams and flood defences, to the buildings in which we work and the ones we call home. Not only has our demand for concrete been unprecedented over the past 200 years with the advancement of technology; it is still increasing today with emerging industrial powers such as India and China creating an ever bigger demand (between 1990 and 2000 production in these countries grew by 55%, whereas in the Western world production grew by a comparatively much smaller 3% [2]). In 2007 around 23.5 million cubic metres of ready-mixed concrete and 38 million tonnes of precast concrete were produced [3] and this is likely to increase due to further demand from developing countries [2]. As the World Business Council for

Sustainable Development stated in March 2010 [4]:

> "Concrete is the most widely used material on earth apart from water, with nearly three tons used annually for each man, woman and child."

Concrete was originally created using water, cement, aggregates and sand with early examples of concrete structures stretching back to as early as 118 A.D., such as the Roman Pantheon which still stands today [5]. With its strength, ability to be moulded into many shapes and relatively low cost, it is easy to see why it is so widespread. As a material concrete is extremely strong under compression but weak under tensile forces which can cause it to crack. To counter this, steel reinforcing bars (rebars) can be placed into the concrete prior to pouring. Other approaches to strengthening concrete include the use of post-tensioned tendons which are inserted into the structure, surrounded by a grout mixture and then tightened to support the structure. As the amount of concrete produced equates to millions of tonnes, the volume of steel that is produced for reinforcement of concrete is also large: in 2006, an estimated 1.2 million tonnes of steel was manufactured worldwide as rebars [6]. Figure 1.1 shows a typical reinforced concrete configuration.

Figure 1.1: A concrete slab with a section of the concrete removed exposing the rebars inside (used with permission from Mike Lion).

1.2 The degradation of reinforced concrete

While the addition of rebars to concrete appears to end some of the problems, it creates a new problem. The corrosion of the rebars within the concrete (discussed in more detail below and Appendix A) can lead to a reduction of the strength of the concrete, resulting in large repair bills and, in extreme cases, partial or total collapse of the structure. The scale of this problem is vast, with costs directly related to rebar corrosion in the US in 2002 at an estimated $276 billion which is roughly 3.1% of the nation's Gross Domestic Product (GDP) [7]. In 2002 the UK concrete repair industry was estimated at a cost in excess of one billion pounds [8]. Perhaps the most striking figure is that out of the total sum spent in the industry, only 47% is spent on new construction, while 53% is spent on repair [9]. This is perhaps due to the large number of buildings that were built during the first half and early second half of the 20^{th}

century which now require assessment and repair due to poor construction practices and/or harsh environmental conditions. This problem is likely to increase in the next 10-20 years as a result of more structures coming to the end of their serviceable lives. From these figures it is clear that corrosion is extremely costly to control in order to keep the infrastructure on which we rely in a serviceable condition.

Concrete as a material consists mainly of high concentrations of soluble calcium, sodium and potassium oxides which once combined with water form hydroxides which, by their nature, are very alkaline [10]. The alkaline environment (typical pH values are 12-13) provided by the concrete should prevent the steel rebars from corroding as the corrosion process is mainly caused by acids. Chlorides can be a cause of corrosion which attack the rebars without a reduction in the pH level of the concrete if present in sufficiently high concentrations. The alkaline environment of the concrete also results in the formation of a passive layer (a thin impenetrable film which probably consists of metal oxides, hydroxides and minerals found in the cement [10]) between the steel and concrete which gives added protection to the steel. Figure 1.2 shows a cross-section of the passive layer interface between a rebar and the concrete. The passive layer also has the added advantage of being able to reform in an alkaline environment should it become damaged.

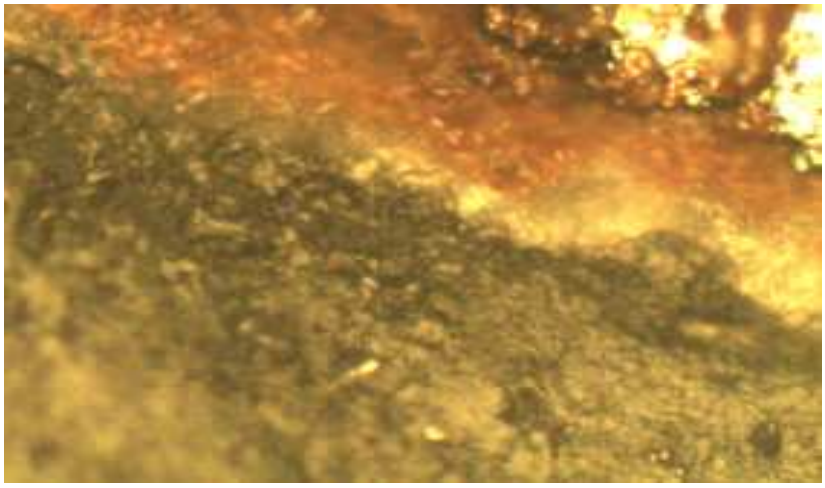


Figure 1.2: A cross-section of reinforced concrete showing the passive layer (dark orange section middle right) between concrete (dark brown section left) and the reinforcing steel (lighter orange area top right). (Used with permission from Boulanger [11].)

Despite the initial environmental conditions of the concrete and the formation of a passive layer, corrosion can still take place after a period of time. This is a result of a reduction in the pH level of the concrete caused by carbon dioxide in the atmosphere. Chlorides can also penetrate the concrete in the form of de-icing salts used to keep concrete structures, such as roads and bridges, free of snow and ice during winter months, and high salt concentrations typically found in marine environments. Poor construction practices, such as insufficient cover levels of concrete and large water:cement ratios exacerbate the problem further as they enable the early onset of rebar corrosion. Insufficient cover has become a bigger problem recently as a result of engineers using as little cover as possible over the last three to four decades for concrete which was not considered structural [12]. This has led to an increase in the number of cases of corrosion of the rebars, where structures built to typically last 30 years require attention and repair after only 15 [12]. Figure 1.3 shows the ceiling of a building where the concrete

has broken away from the structure, leaving the rebars exposed. The probable cause for the spalling of the concrete (where concrete breaks away from the structure) is poor mixing of the cement and the aggregates during construction, which left little cement around the area shown to hold the aggregates together. Over time, these loosely held aggregates broke away from the concrete, exposing the rebars to the atmosphere and increasing their risk of corrosion. Figure 1.4 shows another example of exposed rebars, this time from a lamppost. Here the degradation of the steel is much worse than in Figure 1.3 as a result of increased exposure to environmental factors typical of city street locations. Due to its locality towards the side of the road, it is likely that the poor condition of the steel was exacerbated by de-icing salts.

Figure 1.3: A section of reinforced concrete roof where part of the concrete of the ceiling has collapsed, exposing the rebars.

As a result of the corrosion process, the volume of rust increases causing the rebars to expand which leads to cracking and spalling of the concrete as shown in Figure 1.4. Commonly it is found that for unhydrated ferric oxide its volume becomes double that of the steel; once hydrated, its volume can be up to ten times that of the steel [10].

A detailed account of the two main corrosion processes and their causes, carbonation and chloride attack, is given in Appendix A.

Figure 1.4: A roadside lamppost showing signs of concrete spall and rebar corrosion which was exacerbated, if not caused, by de-icing salts.

Defects such as those shown in Figure 1.4 are often easy to identify from the surface, either as a result of cracked or spalled concrete, or through orange stained areas where water has transported the rust deposits to the surface of the concrete. However, corrosion can also occur without any tell-tale signs to the outside world. In this situation, the anode and cathode where the corrosion processes take place are usually further apart (up to several hundred mm) and there is a lack of oxygen at the anode. This is often the case in marine environments where the concrete is underwater and saturated (see Appendix A for details on the anodes and cathodes, along with the processes which occur at each within reinforced concrete structures). As a result, the iron stays dissolved in the solution, which accounts for the lack of obvious signs of corrosion through cracking, spalling and staining. There are also many other types of corrosion, further details of which can be found in Broomfield [10].

1.3 Structural Collapses

With the huge volume of reinforced concrete that our modern infrastructure comprises, it is little wonder that some structures have deteriorated to the point of partial or total collapse, sometimes even with assessments conducted prior to the collapse. The most prominent danger caused by corroding reinforced concrete is the risk of delaminated concrete falling from the structure which has caused injuries and fatalities. However, even with a small risk of collapse from corrosion, such events can happen. Ingress of chlorides is believed to have caused bridge collapses in Wales in 1985 [13] and Belgium in 1992 [14]. More recent collapses include Pipers Row car park in Wolverhampton, UK [15], where the top deck of the car park partially collapsed overnight in 1997, as shown in Figure 1.5.

Figure 1.5: The partially collapsed top deck of Pipers Row car park in Wolverhampton, UK, 1996 (from Wood [15]).

Another example of corrosion induced collapse occurred in 1985 when a swimming pool roof in Switzerland collapsed, as a result of chloride induced stress corrosion cracking, killing 12 people. Corrosion was accelerated by the humid, chloride-rich atmosphere that is found in many swimming pools. Figure 1.6 shows the collapsed roof of the swimming pool. As stated previously, the number of structures that require attention is likely to increase over the coming 20 years, increasing the risk of further collapses similar to those shown in this section.

Figure 1.6: The collapsed roof of a swimming pool caused by chloride induced stress corrosion cracking (from the Bossard Group [16]).

1.4 Structural assessment techniques

As indicated by the vast costs of repair and the dangers to human life, the accurate assessment of steel-reinforced concrete is of huge importance, but achieving this is often difficult and costly. There are many testing techniques available for engineers to aid in the assessment of structures, ranging from simple visual inspection to ultrasonic and radar surveys, some of which are outlined in this chapter. The automated analysis of this data using techniques such as neural networks is outlined in Chapter 2.

To try to counter the corrosion of the rebars, many testing techniques have been proposed, ranging from simple visual inspection conducted by experts to more sophisticated approaches including ground penetrating radar, electrical impedance and ultrasonics. These testing techniques all have advantages and disadvantages. Where visual inspection is conducted by an expert, it is unable to spot defects deep within a structure; more sophisticated approaches are able to do this at the cost of increased complexity in the data they collect, which can make its interpretation costly and difficult. Some of these approaches use models of the response of a structure over a period of time, while others collect data at set intervals using inbuilt sensors which record the responses of the structure under many environmental conditions. The many structural assessment techniques available fall into two broad categories: destructive testing and non-destructive testing.

1.4.1 Destructive Testing

Destructive testing (DT), as the name implies, involves partially destroying part of a structure in order to gain access to the enclosed reinforcing steel. This allows an engineer to assess the state of the steel in situ. This approach can be costly and is

also damaging to the structure which may have been of sound health. The removal of the encasing concrete can leave the steel exposed to the elements in the atmosphere, causing further problems once it has been re-covered with concrete.

One of the most widely used destructive testing methods is coring. This involves drilling into the concrete and extracting a core of concrete and reinforcing steel from the structure. This approach allows an engineer to see the reinforcing steel in its surrounding environment. However, coring not only has the disadvantages mentioned above, but it is also a very localised testing technique. As some corrosion such as chloride induced corrosion is also localised, healthy extracted cores can be non-representative of a structure which contains multiple defects. The reliability of coring is also affected by the extraction process as some of the physical properties of the core are destroyed during drilling and extraction. This can also give a misrepresentation of the condition of the structure. In addition, coring also breaks the reinforcement inside the concrete which reduces the tensile strength it provides to the concrete structure.

1.4.2 Non-Destructive Testing

Non-destructive testing (NDT) involves using non-invasive techniques for the assessment of a structure's rebars. NDT techniques do not require the removal of concrete and are, therefore, often less time consuming. Additionally, NDT techniques do not damage the structure and can give an accurate assessment of its condition. As a result of their ease of use large sections, if not the whole structure, can be evaluated within a reasonable time-frame. Typical examples of NDT techniques include visual inspection, ultrasonics, x-ray imaging, ground penetrating radar (GPR) and half-cell potential testing. Each of these techniques has advantages and disadvantages. For example, visual inspection is conducted by an expert engineer who is trained to spot defects and

give accurate assessments of their severity, but this approach can be subjective and is unable to detect defects which give no external indications. In many structures the integral parts of the structure are also often hidden, making visual inspection difficult. Half-cell potential is an approach which measures the electric potential across an area of rebars. Electrical potentials within a certain range can indicate problem areas. Depending on the potential recorded, one can make an estimation of the likelihood of corrosion being present according to the American Society for Testing and Materials (ASTM) standard C876-91 [17]. These readings can be influenced, however, by the water content and the electrical resistivity of the concrete. The naturally occurring process of carbonation gives mixed readings as a result of the close proximity of the anode and cathode [10] which can be difficult to interpret. As the focus of this research is the analysis of data from an electromagnetic approach which overcomes the aforementioned problems and is outlined below, the other NDT techniques available are not discussed here in great detail; for a more thorough overview, see the comprehensive report by Strasse [18].

1.4.3 Electromagnetic Anomaly Detection

A new technique has been developed as part of a collaboration between Keele University and SciSite Limited named Electromagnetic Anomaly Detection (EMAD) [19, 20]. Using EMAD a lightweight probe can be wheeled over an area with minimal effort and does not require contact with the rebars. In addition, the properties of the concrete, such as water content, have no effect on the data collected by the probe. This new technique extracts very small electromagnetic signals from corroding steel using the principles of electromagnetic flux leakage. To achieve this the reinforcement mesh must be electromagnetically energised and a suitable probe used to detect its char-

acteristics. One simple way of doing this is to magnetise the steel and measure the ensuing flux leakage. Using a sensitive enough probe and measurements taken at a series of magnetic states a multidimensional dataset can be obtained which contains electromagnetic signatures for both rebar breaks and corrosion. More sophisticated high frequency probes have a higher degree of accuracy which have been investigated thoroughly in a laboratory based environment by Sherratt [21]. However, the current research concentrates on the simple case of DC flux leakage to explain the principle. The EMAD detects electromagnetic signals in the X, Y and Z directions as shown in Figure 1.7 below.

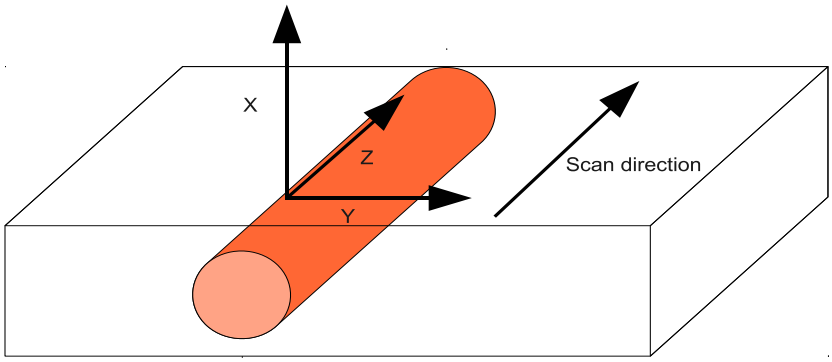
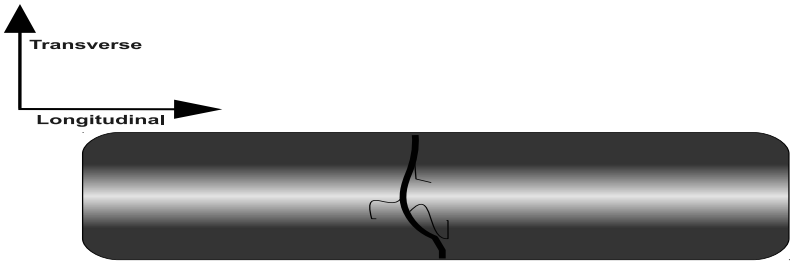


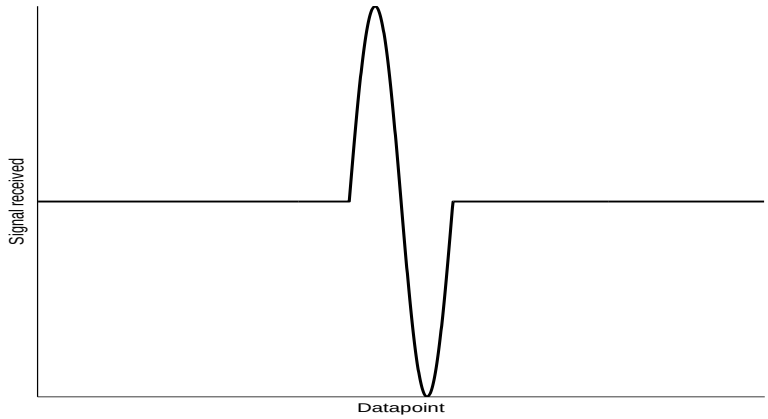
Figure 1.7: The X, Y and Z directions in which the electromagnetic signals of rebars are detected by the EMAD where the scan direction runs in parallel over the rebar shown.

Flux leakage has been used for some time to detect defects such as breaks (for example, Hillemeier and Scheel [22, 23, 24] applied this approach to detect breaks within a reinforcing mesh), but its use for corrosion detection has been somewhat

limited. Flux leakage is based upon the principles of electromagnetism, where a defect, such as a break within the steel, leads to the creation of magnetic dipoles within the mesh, whose magnetic signature appears similar to that near to one end of a bar magnet. Figure 1.8 shows a typical signal created by a break within a rebar. Data collected using flux leakage analysed using backpropagation Artificial Neural Networks (ANNs) have proved successful in the automatic detection of defects within pipelines [25, 26, 27, 28, 29] and also cables typically used in bridges [30]. When analysing data collected from bridge cables a committee (or a cluster) of backpropagation ANNs were used to estimate the size and position of defects within flux-leakage data generated from a computer model which captured the flux leakage signatures of healthy and defective cables typically used in bridges. Using a committee of 51 ANNs an average error rate of 4.5% was found.



(a) A break within a rebar.



(b) The transverse component of the magnetic flux density.

Figure 1.8: A schematic diagram of typical magnetic dipole signal created from a break within a rebar (adapted from Hillemeier and Scheel [31]).

Within a mesh which is of sound health, in the presence of an external magnetic field the magnetic flux seeks paths of lowest resistance. When a defect is present, poles are formed which create a magnetic field which is no longer enclosed in the mesh. It is this *fringing field* which is detected by the EMAD probe. As some corrosion products, such as magnetite, are magnetic corrosion can also be detected by looking

for small magnetic fields produced by corrosion products within the reinforcing mesh. The typical procedure for using the EMAD technique is as follows.

1. A pre-scan of the structure is taken. This gives an indication of the magnetic state of the reinforcing mesh when it has been sitting in the Earth's magnetic field.

2. The structure is energised using a magnet. This puts the mesh into a known magnetic state where defects can be detected.

3. The structure is then scanned again using the probe to detect any differences between the magnetically induced state and the previous, non-magnetised state.

4. Both scans are then compared by calculating their difference.

For most structures the operational approach outlined above gives the most insight into a structure's health. For other structures, such as ones which have been energised by another means such as a stray electric current or vehicles, a pre-scan captures sufficient information removing the need to carry out steps 2, 3 and 4. On the other hand, in scenarios where inspection time is limited, a pre-scan conducted in step 1 may not be collected and only steps 2 and 3 are performed. For all of the data presented later in Chapter 5 no pre-scan was performed; therefore steps 1 and 4 were omitted in the data collection procedure.

A structure can be energised using a number of magnetic field strengths and scanned in multiple directions to give more information on the state of the steel mesh inside it. This is essential for anything other than the simplest of structures. Figure 1.9 shows an image of the probe, energiser and data logger.

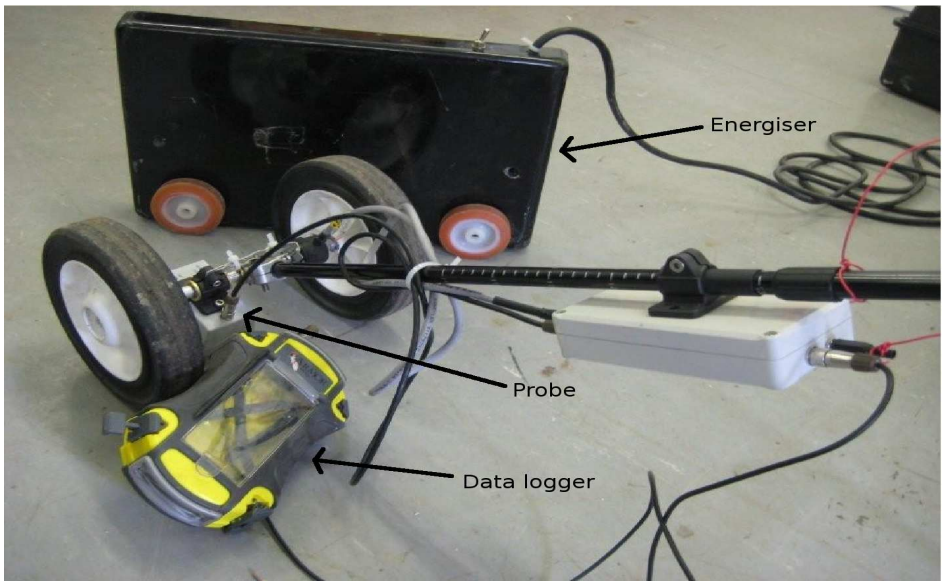


Figure 1.9: The probe, energiser and data logger which make up the EMAD kit.

While the EMAD approach appears in principle to be very simple, in practice this is often not the case. When testing large areas such as motorways and car parks, the amount of data which is produced makes visual data processing very expensive and time consuming. Data is currently analysed by visual inspection of every scan line which is ultimately impractical and undesirable: for large datasets this is very time consuming and expensive as expert analysis is required for the whole dataset.

In order to improve such approaches and speed up the analysis of the data collected by this device, automated data processing approaches can be applied. These range from simple threshold detectors, to expert systems and machine learning approaches where automated detection is achieved using preprogrammed or learned knowledge of defects within particular structures. In order to overcome any errors introduced by creating computer models of a structure's health, as is often reported in the literature (see Chapter 2 for more details), and to provide training and testing data in which the

ground truth, consisting of the location and type, of defects present within the rebars were known, an experimental mesh was created. This mesh contained several types of defects and was encased in concrete. Over a period of time the mesh was subjected to harsh environmental conditions to exacerbate these defects further. This then provided electromagnetic signatures for different defects over a period of time where the type and location of each defect was known a priori: a luxury that is not often afforded when assessing real-world structures.

The fact that an expert can recognise potential areas of defects offers some encouragement that machine learning approaches, which learn by example, can be applied to this problem domain to speed up the detection of defects within reinforced concrete. The research reported here focuses on the application of a machine learning technique, in particular a novel type of recurrent neural network, *Reservoir Computing*, to this domain. Two datasets collected from reinforced concrete structures were analysed, one of which was from the controlled mesh experiment, and the other real-world dataset from an old school building.

One aim of this research is to automate the process and eventually produce a real-time detection system which is able to indicate areas of potential defects while the survey is taking place. (For example, a simple system could give a warning to the engineers informing them that the area they are currently scanning or have recently scanned requires further investigation.) The nature of the data collected by EMAD makes using threshold based approaches to automate the detection of defects difficult as the signals received are signatures rather than voltages: small signals could be a product of a small amount of corrosion near the surface of the concrete or large amounts of corrosion located much deeper. By their nature, the ability of ANNs to

learn and their tolerance to noise makes them well-suited to this task as they can potentially learn to recognise and classify different defect signatures present within a dataset. This has been investigated in the work reported below by training an ANN to recognise areas of potential defects using data collected from an experimental mesh (see Chapter 5) as well as real-world structures using the EMAD. This work can be seen as the first steps towards providing an automated real-time detection system which combines the EMAD with an ANN that is able to recognise potential defective areas based on its knowledge of previous defect signatures. The next chapter outlines the use of ANNs for the detection of defects within the NDT domain.

1.5 Research questions and contributions of this thesis

The research questions this thesis aimed to answer are listed below.

1. How can the trade-off observed in standard Reservoir Computing (RC) architectures (more specifically Echo State Networks (ESNs)), between non-linear separation and short-term memory, be overcome (within the context of highly non-linear time-series datasets)?

2. Does Reservoir with Random Static Projection (R^2SP) overcome this trade-off?

3. Where there are differences in performance between R^2SP, the Time Delayed Extreme Learning Machine (TD-ELM) and other architectures, what are the reasons for these differences?

By answering these questions, the following items are the main contributions of this thesis.

1. A novel R^2SP architecture which combines a standard RC approach (ESN) with two layers of memoryless feedforward neurons, borrowed from the field of ELMs. These two layers perform a non-linear transformation of the input data and reservoir activations which allow the reservoir to be automatically tuned towards a higher memory capacity, improving performance for highly non-linear time-series datasets.

2. When applied to several time-series datasets, overall, the R^2SP is shown to outperform an ESN for the all of the datasets. Through the use of reservoir analysis measures, it is shown that the reservoir of the R^2SP does indeed have a higher memory capacity as a result of the additional memoryless layers transforming the input data and reservoir activations onto a high dimensional state space.

3. Where the R^2SP (and the ESN) are outperformed by the TD-ELM approach, it is conjectured that this is due to the dataset under analysis. Where the TD-ELM is best suited appears to be for datasets whose memory requirements of previous inputs can be captured sufficiently using a hard limited memory approach. In this case, the fading memory of the reservoir may inhibit performance. For datasets where a fading memory is required, a reservoir-based approach (such as R^2SP) is better suited as shown by its superior performance.

1.6 List of publications

Below is a list of publications that have been disseminated through this research, along with the corresponding chapter where further details can be found.

1. Butcher, J.B., Lion, M., Day, C.R., Haycock, P.W., Hocking, M.J. and Bladon, S., A Low Frequency Electromagnetic Probe for Detection of Corrosion in Steel-

Reinforced Concrete, In Grantham, M., Majorana, C. and Valentina, S. (ed.), *Concrete Solutions*, CRC Press, pages 417-424, 2009 (see Chapter 2).

2. Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R. and Haycock, P.W., Extending reservoir computing with random static projections: a hybrid between OP-ELM and RC, In *European Symposium on Artificial Neural Networks (ESANN)*, pages 303-308, 2010 (see Chapters 4, 5 and 6).

3. Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R. and Haycock, P.W., Pruning reservoirs with Random Static Projections, In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 250-255, 2010 (see Chapters 4, 5 and 6).

1.7 Summary

The degradation of the rebars inside reinforced concrete presents a major challenge to much of our infrastructure upon which we depend heavily. Left unchecked, the deterioration of a structure's health can lead to partial or total collapse with serious consequences. The main causes of corrosion, its consequences and methods used to assess a structure's health have been outlined in this chapter, along with the specific data collection technique used in this research. While this technique offers a multidimensional insight into the condition of the rebars inside the concrete without requiring contact with the rebars and, hence, concrete removal, the data it collects can be large, noisy and time consuming to analyse. The findings of two systematic literature surveys of the field of ANNs and structural assessment in engineering found that computational models were widely used, with further details outlined in Chapter 2. While these models can create scenarios which are costly and time consuming to capture in real-world

structures, inaccuracies in a model can affect its reliability when compared to data captured from real-world structures.

As this data is time-series based and requires classification to detect potential defect areas, recurrent neural networks are a good candidate for the automatic analysis of the typical datasets collected from reinforced concrete structures. A recent addition to the area of recurrent neural networks, *Reservoir Computing* (RC), offers the advantages of using standard recurrent approaches, with the added benefit of a simple training approach, where complex training algorithms have often been the downfall of many recurrent neural network approaches, as outlined in Chapter 3. The characteristics of reservoirs and the effect of changing their parameters is also discussed, and it is shown that there exists a trade-off between an ESN reservoir's two main capabilities: non-linear separation via non-linear transformation of the input onto a multidimensional state space; and the ability to possess short term memory, allowing to it recall and learn previous inputs and how they influence the current input.

An extension to RC is outlined in Chapter 4 which combines a standard ESN with two memory-less feedforward layers to overcome this antagonistic trade-off. The properties of this new architecture are then explored empirically by applying it to several datasets, including a controlled experiment consisting of a reinforced concrete slab and a real-world reinforced concrete structure which are outlined in Chapter 5. Using a controlled laboratory experiment overcame some of the disadvantages of using computational models such as finite element modelling to capture the behaviour of a structure under certain conditions with various levels of damage. This allowed the creation of different types of defects whose characteristics and locations were known for the duration of the experiment which were labelled accordingly for ANN assessment.

To enable further analysis of the behaviour of this new extension, other datasets, including several artificial datasets and a real-world speech recognition task are also outlined in Chapter 5. The performance and behaviour of a standard ESN, the newly proposed approach and several other techniques used as benchmarks are outlined in Chapter 6. The analysis of these results is discussed in Chapter 7, while the work is concluded in the final chapter where suggestions for future work are also outlined.

Chapter 2

Computational Intelligence techniques and their application

2.1 Computational intelligence

Of the many approaches used in machine learning, data mining and processing, Computational Intelligence (CI) based approaches are perhaps the most exciting and inspiring. Motivated by processes observed in the biological brain, such approaches have been widely and successfully applied to domains ranging from stock market predictions [32] to brain-machine interfaces [33], speech recognition [34] to robotics [35], and anomaly detection [36] to enhanced document searching [37].

2.2 The brain and its neurons

The brain is, by far, the most complex body organ. With approximately 10 billion neurons and 60 trillion synapses (connections between neurons) in the human cortex [38], this is data processing and parallelism on a grand scale which makes it easy to see

why the functions of the brain are far from yet being fully understood. This problem was perhaps best described by Lyall Watson [39]:

> "If the brain were so simple we could understand it, we would be so simple we couldn't."

A neuron receives input from another neuron or cell (a signal from an optic nerve connected to the eye for example) along a pre-synaptic connection known as dendrites and sends a signal (or output) to other neurons along a post-synaptic projection known as its axon. The difference between the two connections is their shape and distribution; the dendrites have many branches (the name originates from their resemblance to trees) and are irregular in shape, whereas an axon is often much longer, has fewer branches and is smoother in shape [40]. This is because axons play a major role in the transfer of information over larger distances in the nervous system [41]. Other main parts of the neuron include the soma, which is the roughly spherical shaped cell body of the neuron. Figure 2.1 shows the structure of a neuron, while Figure 2.2 shows some Golgi-stained (brain tissue soaked in a silver chromate solution) neurons.

Once a neuron has received an input via its connections and its cell body potential reaches a certain threshold, it releases an output (spike) to other neurons along its axon. The axon is connected to other dendritic trees of other neurons via synaptic connections. A synaptic connection has two sides: presynaptic and postsynaptic, and a synaptic cleft between them. When a signal is sent from one neuron to another, the electrical signal sent from the firing neuron is converted into a chemical one (known as neurotransmitters) which travels across the synaptic cleft. Once at the postsynaptic side of the connection, the chemical signal is converted back into an electrical one. The effect of the received neurotransmitters can either be to excite (increase the cell

membrane potential) or inhibit (decrease the cell membrane potential) the receiving neuron. A detailed overview of neurons, and the different ways to model them and the brain is outside the scope of this research. A thorough overview can be found in Bear et al [41] and Gernster and Kistler [42].

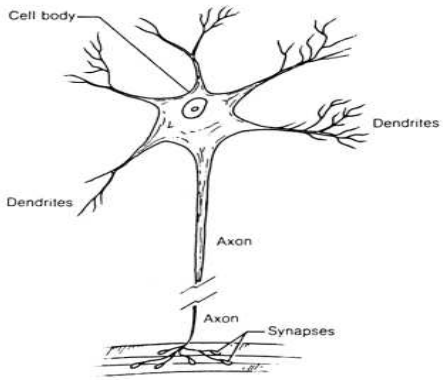


Figure 2.1: A diagram of a neuron (reproduced from Turchin [43]).

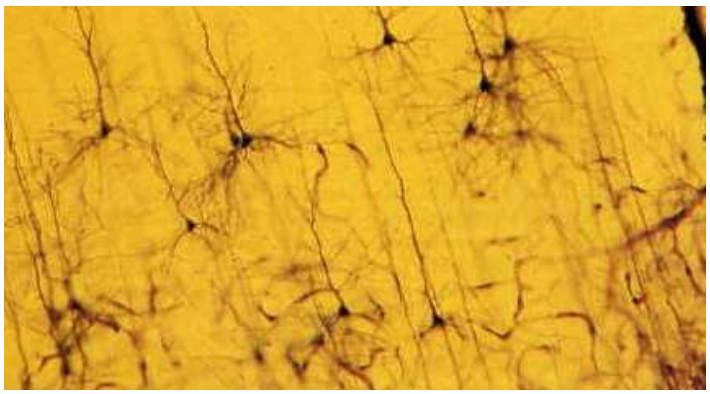


Figure 2.2: An image of some Golgi-stained neurons showing their nuclei, axons and dentrites. The neurons become darkly coloured when stained with a silver chromate solution (reproduced from Hubel [44]).

2.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) (here 'artificial' is widely used to distinguish computational neural network models from their biological equivalent which were briefly presented in the previous section) have become a widely studied and utilised in the field of CI. Taking their inspiration directly from the functionality of biological neurons in the brain, their appeal lies not only in their biological groundings, but also in their ability to learn and recognise characteristics contained within data they are exposed to. Through this learning experience, ANNs are able to recognise and classify, just like the brain, new examples of data they have never seen before. This makes them very appealing to domains where new data is very likely to occur and which requires a fast response (for example, in defect detection). While the human brain is able to learn by adjusting its synaptic connections between it neurons, ANNs adopt a similar approach by altering the weighted connections between each of their neurons during training until a suitable level of output error is reached.

There are many characteristics of ANNs which make them appealing to problem solving in a wide variety of applications. One of their most useful characteristics is their ability to perform non-linear transformations on their inputs: allowing them to deal with non-linear signals which are found in many real-world domains such as speech and sound processing. ANNs also have the capacity to learn the mapping between input data and the target output without the need to program explicit rules into them. This allows them to classify input data correctly using a training algorithm which adjusts the weights of the connections between each of the neurons. Finally, ANNs have the ability to handle data which contain significant amounts of noise.

As no explicit rules are built into the ANNs considered here, once a network is trained it is difficult to analyse which parts of the network are responsible for a particular behaviour. Each trained weighted connection is often meaningless on its own: it is only collectively that these connections perform a task. ANNs are often referred to as black box approaches, as a result of the difficulty of assessing a particular neuron's contribution to the network's behaviour. ANNs can also be prone to overfitting which occurs when the network learns characteristics of noise, rather than being able to generalise over legitimate regularities and variations in data. This makes the network perform poorly on unseen data which is rarely identical to the training data. As ANNs need training on data representative of the problem domain, often large datasets are required in order for the network to learn its different characteristics. This also becomes problematic when using a supervised training approach where the target outputs need to be labelled by the operator in advance which can be costly and time consuming.

2.3.1 Early approaches and the McCulloch-Pitts model

Early work on ANNs began in the 1940's with the creation of the McCulloch-Pitts neuron [45]. A simple McCulloch Pitts neuron, i, receives a variety of k inputs, $\mathbf{u}_k$, via a set of weighted connections with weights w_k which are summed to give z_i. This resulting sum is operated on by an activation function, $\varphi(z_i + b_i)$ to give the output of the neuron, x_i, where b_i is a bias applied to the neuron. Bias is often applied to ANNs in order to shift the activation function of a neuron to a desired location. If the output of the activation function is greater than a certain threshold ϑ, then the neuron gives the output of 1. If the sum is less than the threshold, the neuron fails to fire and its output is 0. These neurons could be arranged into a network to perform a simple logic function.

Variations of the McCulloch-Pitts model have since been introduced which include the perceptron [46, 47, 48] and, later, the multilayer perceptron (MLP) [49, 50, 51]. Training algorithms were also introduced, such as the well-known backpropagation algorithm which changes the values of the weights in order to reduce the output error of the network. A simple one-neuron perceptron is shown in Figure 2.3.

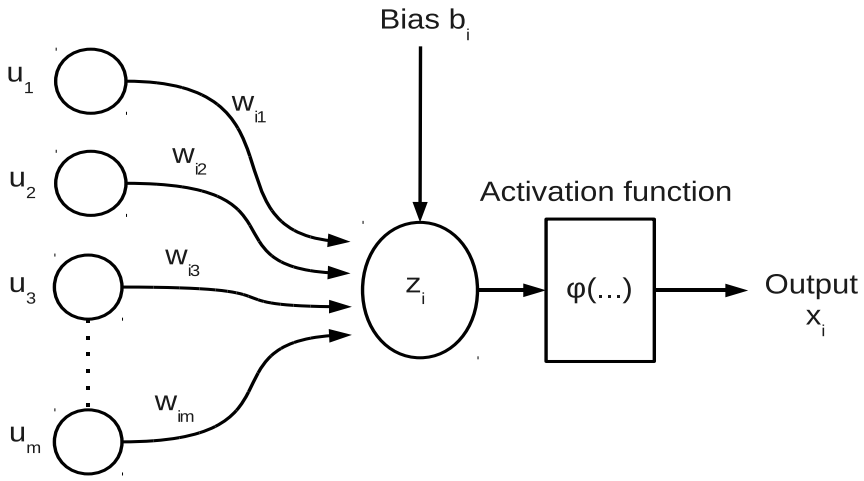


Figure 2.3: A simple perceptron model containing one neuron with m inputs.

A single perceptron is very simplistic and hence its capabilities are somewhat limited since it fails to perform non-linear tasks such as acting as an exclusive OR (XOR) logic gate, as was shown by Minsky and Papert [52]. Towards the late 1980's the work by Rumelhart et al [49, 50, 51] led to revived interest in ANNs as a field. In order to increase the ability of a perceptron to learn, several layers of neurons were connected together to create a multilayer perceptron (MLP) ANN and trained using the backpropagation algorithm which minimised its error using a gradient descent approach. In its most basic form, an MLP ANN consists of an input layer, hidden layer and output layer. The input layer contains neurons in which inputs from a

dataset are presented. The hidden layer contains several neurons which receive input from every input neuron via weighted connections, perform a calculation and output a value to the output layer. The activations of each neuron from one layer feed into the next layer as inputs to each of its neurons. For this reason the MLP and other types of ANNs which feedforward their activations are often referred to as feedforward ANNs.

2.3.2 Recurrent neural networks

The MLP is just one example of the many feedforward ANNs that exist. Since the neurons in one layer send their activations only to the neurons in the next layer, no feedback or recurrent connections exist, and as a result these networks have no memory. These types of networks are, therefore, unable to learn the temporal relationship between input and output datapoints from time-series domains. Recurrent ANNs (RANNs) on the other hand, as the name suggests, contain recurrent connections that are either intra-layer (neurons in one layer feedback into other neurons in the same layer, or themselves), or inter-layer (the output neurons feedback into the hidden layer, for example). These connections feedback the activations of their neurons that have an effect on the output of the network at future time steps [53]. This makes them well-suited to domains containing time-series data where the current input has been influenced by previous inputs, making RANNs well suited to domains including robotics [54], hand-writing recognition [55], weather forecasting [56], power supply load forecasting [57], and speech recognition [58]. RANNs were also shown to be Turing equivalent [59, 60, 61] and universal approximators of dynamical systems, where a network with N outputs can approximately realise a dynamical system of N dimensions [62]. Figure 2.4 shows a schematic overview of a RANN architecture.

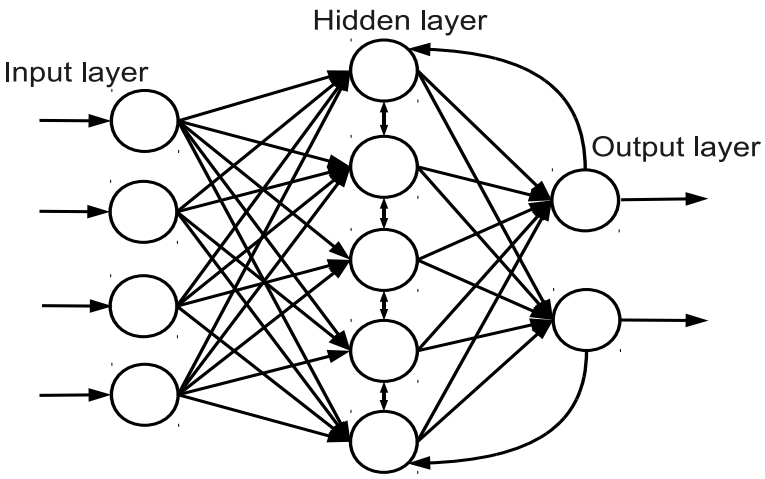


Figure 2.4: A three layer RANN with recurrent connections between the hidden layer neurons and the output neurons which feedback into the hidden layer.

Despite these potentially powerful characteristics, as Lukoševičius and Jaeger argue, the use, and therefore impact, of RANNs has been restricted since their initial inception [63]. The reason for this is that most RANNs are trained using gradient descent algorithms which suffer from several problems such as non-convergence, long training times and vanishing gradients. The problem of vanishing gradients occurs when the values of the gradients often become very small after only a few iterations of training. This makes training difficult and results in a network that cannot cope well with noisy inputs or is unable to learn any correlation between the target output and the recurrent inputs. In extreme cases this makes training almost impossible [40]. To overcome the problems associated with complex and slow training algorithms of traditional recurrent neural networks, a new approach, where only the *output* weights are trained using a *simple linear regression* algorithm, has recently been proposed, named Reservoir Computing (RC). Using this approach, it has been shown that similar, and often

improved performance can be obtained without the complexity of traditional recurrent approaches. Several variations have now been proposed under RC, with the two main techniques called *Echo State Networks* (ESNs) [64] and *Liquid State Machines* (LSMs) [65], which are described in more detail in Chapter 3 (see Lukoševičius [63] for a thorough overview of all approaches). As ESNs were created from an engineering perspective, they are the main focus of this work, although LSMs and their similarities to ESNs are discussed briefly in Chapter 3.

2.4 Neural Networks for Structural Assessment

A neural network's ability to learn some of the characteristics contained in datasets and classify them accordingly makes it very appealing for the automatic detection of defects within reinforced concrete. Over the last 20 years or so, ANNs have been widely applied for this purpose in many civil engineering domains, often giving good performance. According to Flood [66], the popularity of ANNs in the NDT domain has been high over the last 15 years, where 12% of the articles in the American Society of Civil Engineers (ASCE) journal [67] have applied ANNs. He also argues that ANNs have slowed in their development and asks the question of why the use of ANNs in NDT has been so widespread if their development has stagnated, to which he answers, "...despite their present rudimentary form, (ANNs) are very good at solving direct mapping problems that are non-linear and comprise several independent variables, a common class of problems for engineering" [66]. In studies which compared ANNs to the state-of-the-art they have been generally reported to offer at least comparable performance, in some cases giving improved performance over other classification techniques.

During the initial stages of this research project a mapping study [68] was conducted with the aim of systematically assessing the use of ANNs within the domain of NDT of structures. This was conducted as a precursor to a Systematic Literature Review (SLR) [69] both of which provided a repeatable, objective approach for assessing literature from this domain. (SLRs have been used extensively in fields such as medicine [70] and have recently also been applied to software engineering [69].) The mapping study provided a broad overview of the use of ANNs for structural assessment with the aim of then focusing the SLR which would be conducted at a later date. The findings of the mapping study showed that little literature existed on the use of ANNs for corrosion detection in reinforced concrete, so the SLR was widened to analyse the use of ANNs in the engineering domain in general. The findings from both the mapping study and the systematic literature review are discussed in the remainder of this section (for the mapping study report, see Appendix B). Some of the ANN techniques used in the identified literature are also discussed, especially where these techniques are used as comparators later on in this thesis.

2.4.1 Feedforward Neural Networks and backpropagation

Analysis of the types of ANNs used in NDT assessment revealed that the most widely used ANN is a feedforward, memory-less MLP network, often trained with the backpropagation algorithm. For example, from the mapping study a total of 179 relevant papers were identified, of which 95 used feedforward neural networks, with 87 of those using either standard MLPs trained using backpropagation or a modified version (using the Levenberg Marquardt (LM) algorithm [71, 72] for example). In one such study, Pratt and Lawrence [73] applied a backpropagation-based ANN to interpret data collected from impact-echo testing of concrete structures collected from

laboratory experiments and numerical simulations conducted by Sansalone and Carino [74]. The concrete structures contained several simulated voids. The ANN was set the task of estimating the probability of a void being present and the depth of the void. The trained network was then tested on unseen samples where an accuracy of 90% was achieved, although the network confused the location of defects when the sensor was placed close to the edge of a void.

Other studies that used backpropagation in the domain of NDT include: the classification of vibration signals from a variety of structures including bridges and prestressed concrete beams [75, 76, 77, 78, 79, 80, 81, 82, 83, 84]; the prediction of expert ratings of the condition of bridges using a set of bridge-state parameters as inputs [85]; the classification of transmissibility vibration data collected from an experimental steel frame [86]; the classification of half-cell readings taken from concrete slabs [87] and piezoelectric sensor data collected from a quarter-scale bridge section [88]; the estimation of the compressive strength and/or slump of concrete [89, 90, 91, 92]; various NDT approaches to locate rebars within reinforced concrete [93, 94, 95, 96]; the diagnosis of composite materials [97, 98]; and the classification of data created from several finite element models (FEMs) and parameterised models [99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115].

2.4.2 Modified backpropagation approaches

Modifications of the backpropagation algorithm have been applied to the NDT domain ranging from the LM algorithm [116, 117, 118, 119, 120, 121, 122, 123, 124], backpropagation with a tunable steepest descent algorithm for the assessment of cantilever beams based on their loss of stiffness [125], backpropagation with an adjustment factor in each modification iteration [126] and the use of the resilient backpropagation

algorithm [127] to detect crack damage in an eight storey frame modelled as a reduction in its stiffness [128].

2.4.3 Other feedforward Neural Network approaches

Other types of feedforward ANNs investigated in the NDT domain include: Probabilistic ANNs (PANNs), in which the output of the network is determined by a probability density function which estimates the most probable class to which an input pattern belongs, for the detection of damage within an experimental steel structure [129] and a computer model of a bridge [130]; radial basis function (RBF) ANNs, whose hidden layer neurons' activations typically follow a Gaussian distribution, which were used to detect defects within a signal [131] and classify data from several simulations [132, 133, 134, 135, 136, 137]; a statistical ANN which was able to account for the errors and uncertainties introduced by computer models to detect damage within a reinforced concrete slab FEM [138]; and an adaptive resonance theory ANN for the classification of simulated vibration data of a sea platform structure [139].

It could be argued that the popularity of the feedforward ANNs and backpropagation in many studies is due to their use as black boxes. The fact that feed-forward MLP ANN and back-propagation were used so extensively could also be due to their simplicity and ease of use, as Adeli [140] points out. This point is also mirrored by Flood [66] who claims that ANNs place less strain on the user who requires little knowledge of the technique being used. Ease of use, however, may not be the best characteristic to rely on when choosing a type of ANN or alternative method to perform data analysis. For instance, the back-propagation algorithm has many shortcomings, including slow learning times and a convergence rate which is highly dependent on the learning and momentum parameters chosen [140]. If these parameters are chosen poorly, the ANN

can get stuck in local minima which impacts negatively on the convergence rate and the performance of the ANN. For these reasons some of the studies' results could have been improved had a more suitable ANN or alternative technique been used.

2.4.4 Unsupervised Neural Network approaches

While supervised (where a target output is provided) feedforward ANNs were identified as the most popular ANN approach in the NDT literature, unsupervised approaches have also been investigated (where no target output is provided, instead the network learns to distinguish datapoints from different classes by recognising their underlying characteristics). Perhaps the most famous unsupervised approach is Kohonen's Self-Organising Map (SOM) [141]. SOMs are particularly suited to noisy datasets whose output clusters are difficult to determine prior to training. Their topological ordering properties mean that the network is able to place an input pattern between two (or more) clusters when its characteristics suggest it might be a mixture of those classes. This property is not easily achieved using the standard MLP networks discussed previously.

Meksen et al [142] applied a SOM to cluster defects in data collected from ultrasonic inspection of welds. Other uses of the SOM in the NDT domain include: defect detection within data collected from a real-world car park [20]; the clustering of fast Fourier transform (FFT) acoustic emission signals containing several defects in various case studies which mapped examples of uniform corrosion, pitting and stress corrosion cracking into well formed clusters [143]; and the study by Sun [144] where input data from a steel box girder was first transformed using principle component analysis (PCA). A trained SOM was able to classify all cases of single area damage and worst error rate in multiple damaged areas was 30%. Two other unsupervised approaches (a

Probabilistic Resource Allocating Network [145] and a DIGNET network [146]) were investigated by Yeung and Smith [147] to classify data created from an FEM model of a suspension bridge where a damage identification rate of 70% was achieved.

2.4.5 Hybrid approaches

Combinations of several architectures and algorithms, creating a hybrid approach, have also been investigated in the domain of NDT. For example, continuous corrosion monitoring using acoustic emission was proposed by Wevers et al [148] which aimed at detecting three corrosion processes: uniform corrosion, pitting and stress corrosion cracking (SCC), which was the cause of the collapse of the swimming pool roof presented earlier in section 1.3. Once the input data was transformed into the frequency domain, PCA was performed to extract the most important parameters which were then used as input to a SOM to estimate the probability density function where it clustered different types of corrosion well. The data was also applied to a Least Squares Support Vector Machine (LS-SVM) [149] (see Chapter 3 below for a brief overview of SVMs) for a quantitative assessment of separability which involved two tasks. The first task was to detect uniform corrosion from pitting and SCC, while the second task was to distinguish pitting from SCC. The LS-SVM achieved close to zero error for the first task, and roughly 1% error for the second task.

Genetic algorithms (GAs) [150] have also been combined with ANNs to form hybrid approaches, most of which are used to evolve the weights of the networks. GAs are inspired by natural evolution where the fittest subset of a population create new offspring (i.e. weights) which can be combinations of the chosen subset. The less fit members of the population are removed. This process can repeat for many generations, so that each subsequent population should become better suited to solving the prob-

lem. Uses of GAs in NDT include a PANN (see this section above) trained using a GA to diagnose pre-stressed concrete piles [151]; the use of an ANN and GA to predict the performance of bridges using parameters such as geometry, age and traffic loads where the ANN was found to perform better when using different models based on subsets of the data, while the GA outperformed the ANN on the whole dataset, although it was much slower to train [152]; several fuzzy expert systems using genetic algorithms and ANNs for damage detection of reinforced concrete bridge decks [153]; a GA trained ANN combined with information fusion and Shannon entropy [154] to locate damage in an FEM model of a bridge [155] and a GA trained ANN to detect crack width in reinforced concrete which was shown to outperform a backpropagation ANN [156].

Other examples of hybrid approaches include: the use of an MLP with backpropagation, a SOM and Learning Vector Quantisation (LVQ) [157] to detect flaws present within steel plates [158]; an ANN with case-based reasoning (CBR) and a Monte Carlo Simulation (MCS) to predict the onset of corrosion [159]; an ANN coupled with Latin Hypercube Sampling (LHS) applied to a simulation and two experimental concrete test pieces where it was shown to reduce training times significantly and improved the identification of the parameters of the model [160]; an ANN with an integrated fuzzy interpretation of various levels of damage of an FEM modelled prestressed concrete bridge [161], an experimental model of a steel bridge which can be used on any type of structure [162] and an experimental reinforced concrete slab with a break in the rebar where average detection rates of 98% were found [163]; the use of a fuzzy neural network and a data fusion technique to classify structural vibrational responses [164]; a PANN combined with a backpropagation ANN to detect defects within a simulation of a structure under various damage conditions [165]; and the estimation of defect location through loss of stiffness of a cantilever beam which was analysed using a fuzzy

cognitive map (FCM) combined with an unsupervised Hebbian learning approach [166]. In the last study it was found that this approach gave errors of 2% in the presence of noise. A slightly different approach was adopted by Lee et al [167] where a fuzzy Petri net for the classification of bridge data was investigated which offered improved rule-based reasoning.

2.4.6 Data collected from computational models of structural defects

Analysis of the literature also reveals a shortage of analysis of data collected from real-world structures, with the majority of studies using computational model approaches or laboratory experiments as the source of the data to be classified (for example, out of the 179 papers identified by the mapping study, 79 used simulated data which was closely followed by 78 studies using laboratory experimental data). One potential problem with using computational models to observe the behaviour of a structure under certain conditions is the inaccuracies introduced into the model by the investigator. Although some approaches have been proposed to overcome this (for example, the work conducted by Bakhary et al [138] presented above in section 2.4.3), it is very difficult to define and capture the characteristics of a structure accurately. A model also needs to be created for each structure individually which may not cover all possible defect scenarios [168]. The popularity of model based approaches probably comes from their relative ease to set-up and the relatively short time it takes to create a defect within a model of a structure when compared to a laboratory experiment or a real-world structure where it can take many months (or years) for defects to occur. Simulations also have the added advantage that defects or anomalies can be created in the dataset which are often rare in real-world data (strain data from a bridge for

example can contain many years of data during which no anomalies are present) [168].

2.4.7 Data collected from laboratory controlled experiments

The use of experimental data overcomes any inaccuracies introduced by model based approaches by replicating as closely as possible the conditions of a real-world structure containing defects. Controlled laboratory experiments also have the advantage of added insight into the condition of the structure below the surface as the investigator usually knows where defects are and what type of defect are present as they are usually designed into the experiment. The downside to experiments is the time it takes to recreate defects found in the real-world which can often take up to 30 years to become problematic and that the defects and/or conditions created may not be true examples of real-world scenarios.

2.4.8 Data collected from real-world structures

The reason that so little data from real-world structures is present in the literature is probably due to the fact that most data from real-world structures are collected by industrial companies who specialise in conditional assessment. This data may be commercially sensitive either to the contractor or the client. Real-world data is also expensive as it involves specialised data collection, usually by a team of experts. The major advantage of real-world data comes from the fact that it captures the condition of the reinforced concrete in-situ where potential defect areas in the data can be verified if necessary with destructive testing techniques. The data can, however, depending on the concrete removal process, be difficult to verify against the actual condition of the structure, since even when destructive investigations are conducted the condition of the structure often changes during the extraction process.

2.4.9 The use of Recurrent Artificial Neural Networks

Perhaps one of the most surprising findings from surveying the literature is the apparent little use of recurrently connected ANNs in the NDT domain. Through their setup, RANNs possess memory which allows them to learn temporal characteristics contained within a dataset. This ability to remember previous inputs and how they influence the current input makes them well suited to time-series datasets as discussed earlier and so the potential use of RANNs in such domains, including the NDT domain, is highly appealing. Many of the datasets collected in the NDT domain are time-series based, such as the EMAD introduced earlier in section 1.4.3, making RANNs an ideal candidate for analysing and detecting potential defect areas. The majority of studies that used RANNs in the NDT domain used a time-delayed version of the input data, effectively creating memory through the size of the window of data selected. The problem with this approach is that the network's memory is only equal to the size of the window of data, any data outside this window the network will not remember. The size of the window used therefore becomes another parameter to optimise.

Despite this, several studies have used a time-delay approach. Barai and Pandey [169] and Niu [170], used a delay at the input layer to introduce a time scale within the network, giving it memory which was equal to the delay. Another approach used to capture some representations of time-series data was presented by Kao and Hung [171] who investigated the use of an adapted feedforward ANN named neural system identification network (NSIN) trained using the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) learning algorithm [172] on simulated and experimental structural vibrational response data which was sampled over a given time period. This approach was found to offer low error rates for various scenarios while converging faster

than a backpropagation algorithm.

Huang and Lo [173] also investigated an approach using delayed inputs to an LM trained ANN to classify seismic data collected from a simulation and a real-world bridge structure. Another delayed input approach was also investigated by Hung et al [174] where wavelet ANNs (WANNs) [175] were used to model the behaviour of an experimental five-storey half scaled steel frame under several earthquake strength scenarios. An MLP trained with the LM algorithm was compared to a WANN where both offered similar performance, although the WANN offered superior convergence time. The use of an Auto-Associative ANN (AA-ANN) applied to data collected from bridge sensors was investigated by Lim at al [168]. In this study an MLP configured as an AA-ANN was trained with backpropagation to predict the strain of a bridge using real-world data collected in the past, where any deviations from the predicted strain and the actual strain were perceived as anomalies in the structure. Delayed versions of the input were presented to the network as this was found to improve the detection of anomalies, which the network was able to detect accurately.

The problem of using a moving window representation of the input can be overcome when using a network with recurrent connections where the memory of the network can potentially stretch further into the past to data points that, in the time-delayed version, would be outside of the moving window. In one such study, Jiang and Adeli [176] investigated the use of a multi-paradigm time-delay fuzzy WANN to capture the behaviour of two simulated high-rise buildings when subjected to various degrees of earthquake tremors. Here recurrent feedback loops were used to give the network memory and allow it to map the relationship between the previous and current inputs and outputs. WANNs have also been applied to modelling the behaviour of a dam

in which rough set theory was applied to remove redundant neurons according to the dependency of the output of the network and the neuron in the hidden layer, where it was shown to outperform a wavelet network proposed in a separate study [177] and a backpropagation ANN reported in [178]. In another study, a Hopfield network [179] was applied to the restoration and enhancement of NDT images by Chen [180] who compared it against other approaches which included independent component analysis (ICA). According to Chen [180] all of the approaches were found to give similar performance.

Perhaps one reason that RANNs have seldom been used in NDT applications could be the complex and slow training procedures often associated with RANNs. In the next chapter a recent addition to the field of RANNs named Reservoir Computing (RC) offers an attractive solution to this problem but, as of yet, has not been applied to automatic detection of defects, more specifically corrosion and breaks, within reinforced concrete structures. Furthermore, no RC analysis of EMAD data collected from laboratory controlled and real-world scenarios has been conducted so far, which this current work reports on the results of doing this in Chapter 6.

2.5 Summary

This chapter has introduced Artificial Neural Networks (ANNs) as a Computational Intelligence approach which can be used, as shown by their extensive use in the field of NDT, for the analysis of data collected from NDT techniques. With their ability to learn the relationship between an input pattern and an output pattern by example, tolerance to noise, and perform non-linear transformations of the input data, ANNs are well suited to data collected from NDT domains. ANNs have been applied to

a wide range of NDT applications, ranging from the classification of defects present within bridges and pre-stressed concrete beams to the location of rebars within concrete and the classification of data from various models of structures. Recurrent ANNs (RANNs), by their nature, have a short-term memory, and are therefore able to learn the relationship between input and output data over a period of time. This makes them well-suited to time-series data where the current input/output may be a result of a number of previous input/output data. This is a characteristic of much of the data collected from the NDT domain.

There also appears to be a total lack of research into the detection of corrosion within real-world reinforced concrete structures, with no publications found during two systematic surveys of the literature. The most popular data generation approach is the use of models to generate the response of a structure under certain conditions. While this approach can generate lots of data, the data generated may not be a correct representation of the same structure in the real-world. To overcome this, a laboratory controlled reinforced concrete mesh experiment is proposed to allow the collection of real-world data which contains the known type and location of different defects, where further details are given in Chapter 5.

The use of RANNs in the NDT domain has been somewhat limited even though they would appear to be best suited for the analysis of time-series NDT data. This is perhaps due to the complex and time-consuming nature of the training algorithms that many RANN approaches use. Not only are these training algorithms slow and complex, but are often susceptible to local minima and vanishing gradients which can hinder performance further. This may also offer an explanation as to why feedforward approaches, often trained using backpropagation, are the most widely used ANN in

NDT data analysis, as these approaches are much simpler, faster and, as a result, give better performance. An alternative approach to training RANNs which is efficient, fast and has been shown to at least perform as well as the state-of-the-art, Reservoir Computing, may be well suited to the automated classification of EMAD NDT data, which is outlined in the next chapter.

Chapter 3

Reservoir Computing

3.1 Introduction

Reservoir Computing (RC) emerged ten years ago as a promising new area of the field of Recurrent Artificial Neural Network (RANN). Of the different models which RC is comprised, Echo State Networks (ESNs) [64] and Liquid State Machines (LSMs) [65] first originated around ten years ago and are perhaps the most widely studied and used. Other techniques have since been added to the field of RC including the Backpropagation-Decorrelation (BPDC) learning algorithm introduced by Steil in 2004 [181]. As a result of their recurrent connections, RC techniques are well suited to time-series data where the current input and/or output depends on a number of previous inputs and/or outputs. The use of reservoirs for time-series prediction was one of their first applications for which they offered improved performance over other approaches [64, 182, 183, 184] (see section 3.8 for more details on the applications of RC).

Although developed independently, ESNs and LSMs share very similar characteristics. They are both based on a reservoir of randomly and sparsely connected neurons

which can be trained by a simple linear regression technique [185]. RC approaches therefore separate the task of mapping the input onto a higher dimensional state space (where non-linear data can be separated) and training the linear, non-recurrent readout which draws hyperplanes in the expanded state space to produce the target output.

All RC implementations require only the readout (output) weights to be trained. All other weights are fixed from the point of network creation and initialisation, therefore overcoming the problems inherent in typical recurrent networks such as slow convergence or even non-convergence and slow and complex training procedures [186]. Reservoir approaches also have the advantage of possible extension to other problem domains. For example, in classification tasks extra output units can be added to the network to accommodate any data which belongs to a new class. As the output weights are independent of each other, only the newly added weighted connections need to be calculated, avoiding a phenomenon known as catastrophic interference [187] which refers to the interference of newly added problem characteristics on the previously learned weighted connections of the network.

From an alternative viewpoint, RC can be thought of as a kind of temporal kernel method as the reservoir performs a transformation of the input onto a high dimensional state space. Kernel approaches are well studied in machine learning. In this technique one can transform data onto a feature space using the inner product of the input data in order to separate different classes linearly. Briefly, a kernel K can be defined as $K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$ where $\mathbf{x}$,$\mathbf{z} \in U$, U is the input data and ϕ is the mapping from U to feature (state) space [188]. Perhaps the most well known kernel approaches are Support Vector Machines (SVMs) [189] which use the kernel trick [190]. The kernel trick overcomes the need to explicitly calculate the computationally expensive inner

product of the input to project it onto the high-dimensional state space (ϕ) using a kernel function.

While SVMs use the kernel trick to transform the input data onto the state space, in RC, the reservoir is used to perform the transformation which is non-linear and also spatio-temporal as a result of the reservoir states which have been influenced by all of the previous inputs through their recurrent connections. Standard SVMs are not recurrent however, and differ from standard RC techniques in the sense that they have no memory. One approach for providing an SVM with memory is the use of a moving window or delayed input approach which was discussed in the previous chapter. While this approach allows an SVM to contain some memory regarding previous inputs, the memory only extends as far back as the window width with an abrupt end to memory of previous inputs outside of the window, giving it a kind of 'hard' memory representation. A reservoir on the other hand has a 'softer' memory of previous inputs as a result of its recurrent connections that provide it with a fading memory extending through time.

Several recurrent SVMs have also been proposed removing the need for the sliding window of inputs. Examples include the Recurrent Least Squares SVM (RLS-SVM) [191], the multidimensional recurrent SVM (MRLS-SVM) [192], the intrinsic recurrent SVM (IR-SVM) [193] and an EVOlution of systems with KErnel-based outputs (EVOKE) [194]. The RLS-SVM approach used a time-window of previous output values which are fed back into the network as inputs where it was shown to perform well on a difficult trajectory learning task [191].

EVOKE [194] combined a relatively novel type of RANN as a preprocessor to an SVM. This RANN approach, named EVOLution of recurrent systems with Optimal

LINear Output [195] (EVOLINO) is based on Long Short-Term Memory (LSTM) models where the weights of the network are evolved. This approach was shown to offer good performance on a grammar learning task and a multiple superimposed sinewave task which is difficult for other approaches to solve. For example, single reservoirs find this task difficult as a result of their coupled neurons (coupled refers to the fact that the reservoir neurons are connected to each other and are therefore unable to be trained to generate multiple sinewaves with varying oscillations [196]).

The IR-SVM [193] incorporated recurrency directly within the SVM optimisation problem and as a result was able to identify underlying dynamical systems of two tasks involving the summing of previous inputs to give a desired output and the prediction of superimposed sinewaves with promising results.

In the work by Sun et al [192], the MRLS-SVM was proposed and was shown to achieve noise reduction and function approximation for chaotic time-series with accurate results.

3.2 Liquid State Machines

The main difference between the LSMs and ESNs derives from their initial origins and concerns the properties of their neurons and their activation functions. LSMs were created from a neuroscience perspective and, hence, are more biologically motivated and, as a result, their liquid (or reservoir) contains more biologically plausible neurons than ESNs. They are usually implemented using spiking neurons to create neural microcircuits. Spiking neural networks share more similarities to their biological equivalent as they are based on spike trains (sequences of outputs, or spikes). Other

characteristics of spiking neurons include some built in internal memory and the inability to spike during or immediately after a recent spike. The neuron can only spike again after a period known as the absolute refractory period, after which a period of relative refractoriness occurs where it is difficult for the neuron to spike.

As the spike of each neuron is the same, it is often (although other approaches can be used) the firing rate (based on the number of spikes over a time period) or the time a neuron spikes that are of interest. Perhaps the most widely known spiking neuron model is the leaky integrate and fire (LIF) neuron [42]. In this model, any inputs the neuron receives leak away over time (hence the term leaky) if an insufficient amount of inputs are received and the neuron's threshold is not exceeded. If sufficient input is received to exceed a threshold, the neuron outputs a spike. Although these types of neurons are very interesting from a biological perspective, a detailed discussion and investigation is outside the scope of this research. Further details can be found in Maass and Bishop [197], or Gerstner and Kistler [42], as well as in Schrauwen [198] where the use of spiking neurons in engineering applications is investigated thoroughly.

As the spiking neurons used in LSMs are more complex than the standard McCulloch-Pitts sigmoid neurons [45], they have been shown to be more computationally powerful than the sigmoid neurons commonly used in ESNs [199]. As a result fewer spiking neurons are usually required to give the same performance as a network containing analog (for example, sigmoid) neurons. LSMs have, however, been somewhat restricted with regard to practical applications as a result of too many conditions being required when constructing the reservoir [200] which makes it difficult to obtain a reservoir with the desired characteristics, whereas ESNs have a much broader parameter range over which they can perform well [201]. Although both implementations are biologically

inspired, ESNs were developed for signal processing and control applications [186] and are the main focus of this work. In the remainder of this thesis the terms reservoir and ESN will be used interchangeably unless otherwise stated.

3.3 Echo State Networks

Developed by Jaeger [64, 184], a typical ESN's architecture consists of three layers: an input layer, a reservoir layer and an output layer. Every neuron of the input layer has weighted connections to every neuron in the reservoir layer. (Optional direct connections to the output layer are also possible.) Inside the reservoir layer, recurrent connections between each neuron are present. Each reservoir neuron also has weighted connections to every neuron in the output layer. The output neurons can have optional recurrent feedback connections to the reservoir neurons and recurrent connections to themselves and the other output neurons.

Figure 3.1 shows a typical ESN configuration. Note, only the connections of the first input, reservoir and output nodes are shown for simplicity. In the work presented here no input-to-output, output-to-reservoir or output-to-output connections are used but are shown in Figure 3.1 for completeness.

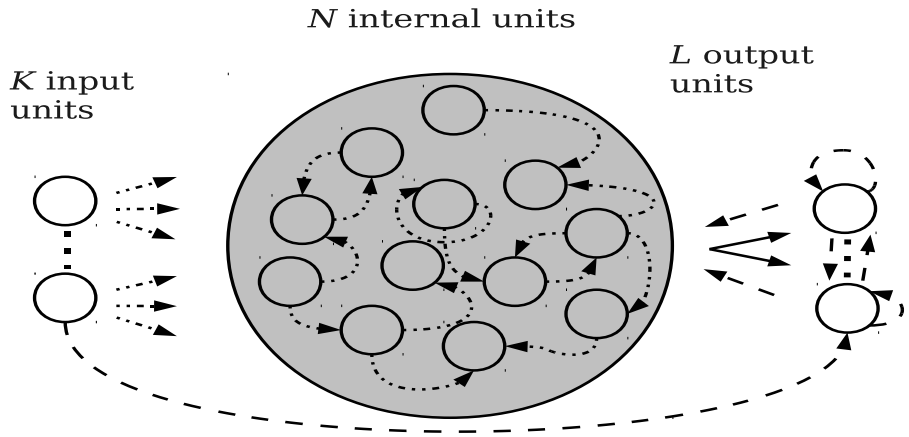


Figure 3.1: A schematic representation of an ESN showing the feedforward and recurrent connections (finely dotted lines indicate weights which are left untrained after initialisation and rescaling, solid lines indicate the weighted connections which are modified during training and thicker dotted lines indicate optional connections which were not used in the work presented here).

3.3.1 Reservoir creation

At creation, an ESN's connections are given random weights, usually from a normal distribution, and a connection sparsity. Connection sparsity refers to the number of connections between neurons which are zero weighted and can, therefore, be considered non-existent. Reservoirs are often created with around 10% non-zero connections, although for *tanh* neurons the connectivity of a reservoir does not affect its generalisation capabilities significantly due to the continuous nature of their activation function [201]. This observation was also recently reported by Gallicchio and Micheli [202]. A reservoir implemented using spiking neurons, on the other hand, was shown to be highly dependent on the connectivity of the network, where performance was optimal with topologies where either a fully connected network with a low fan-in connection scheme,

or a sparsely connected reservoir with high fan-in connection scheme (as is observed in the brain where a high fan-in roughly between 10^3 and 10^4 has been observed [203]) were used.

3.3.1.1 Commonly used reservoir activation functions

Reservoirs of the ESN type typically contain neurons whose activations are calculated based on the sigmoid (or more specifically *tanh*) activation function. The *tanh* activation function is calculated by:

$$tanh(\mathbf{z}) = \frac{2}{(1 + exp(-2\mathbf{z})) - 1} \tag{3.1}$$

Other activation functions can also be used within a reservoir, including linear, threshold and Fermi activation functions. The Fermi activation function is a type of sigmoid activation function which has often been used in reservoirs where *intrinsic plasticity* (IP) has been applied (see section 3.3.2.4 below). The Fermi activation function is calculated according to:

$$fermi(\mathbf{z}) = \frac{1}{1 + exp(-\mathbf{z})} \tag{3.2}$$

Figure 3.2 shows the linear, threshold, tanh and Fermi activation functions.

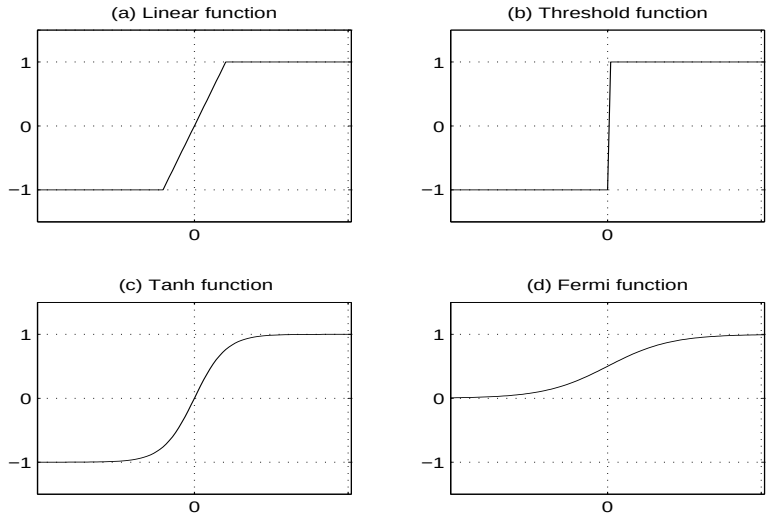


Figure 3.2: The linear, threshold, tanh and Fermi activation functions commonly used in neural networks.

3.3.1.2 Datasets commonly investigated in the Reservoir Computing literature

Within the ESN literature there are several datasets which are often used as benchmarks and for investigative studies. The majority of these datasets are beyond the scope of this research, although two datasets are referred to in the remainder of this chapter and hence are briefly outlined below.

The Mackey-Glass task is a well-studied task in the RC field for which a reservoir is required to generate the input based on a time delayed differential equation without any external inputs which is achieved by training the reservoir in free-running mode. Once trained the output of the reservoir is fed back into the reservoir instead of the teacher forced signal. Often delays of 17 and 30 time steps are used where a value of 30 makes the task difficult.

The second dataset that is often used and is reported below in the Non-linear Auto Regressive Moving Average (NARMA) dataset. The NARMA task was introduced in [204] and is a non-linear system identification task where, in the 10^{th} order case, the current output is influenced by various proportions of the last 10 inputs and outputs, which can be difficult to model.

3.3.2 Reservoir training

As mentioned previously in section 3.1, the main advantage of RC techniques is the relative simplicity of their training. Training of a typical RANN requires every weighted connection to be changed many times in order to reduce the error of the output layer. This process can be very long and computationally expensive. After the difficult process of choosing the correct parameters and a computationally intensive training regime, the network weights obtained may give sub-optimal performance as typical RANN training consists of gradient descent approaches which are susceptible to local minima. Vanishing gradients are another problem associated with gradient descent algorithms when training RANNs, introduced in Chapter 2, which can make training difficult or almost impossible, giving a network that cannot cope well with noisy inputs or is unable to learn any correlation between the target output and the inputs [40]. Due to these complexities, the use of RANNs in real-world problem domains has been somewhat limited, despite their initial promise [63].

Training of an ESN and other RC techniques, however, requires only the weighted connections between the reservoir and the output layer to be modified and is, therefore, much faster. The trained weights of a reservoir are also optimal as they are calculated using a regression technique based on the Ordinary Least Squares (OLS) approach which finds the global minimum of error, overcoming the problems associated with

gradient descent approaches and local minima. For ESNs the activation of reservoir neurons at time t is calculated according to:

$$\mathbf{x}(t) = f(\mathbf{W}_{res}^{inp}\mathbf{u}(t-1) + \mathbf{W}_{res}^{res}\mathbf{x}(t-1) + \mathbf{W}_{res}^{out}\mathbf{y}(t-1) + \mathbf{W}_{res}^{bias}) \tag{3.3}$$

where $\mathbf{x}(t)$ is the activation of the reservoir's neurons at time t, f is the reservoir neuron activation function (usually *tanh* for ESN reservoirs), $\mathbf{W}_{res}^{inp}$, $\mathbf{W}_{res}^{res}$ and $\mathbf{W}_{res}^{out}$ are the weights of the input-reservoir, reservoir-reservoir and output-reservoir connections respectively, $\mathbf{u}(t-1)$ is the input neurons' activation vector at time $t-1$, $\mathbf{x}(t-1)$ is the reservoir activation vector at time $t-1$, $\mathbf{W}_{res}^{bias}$ is the bias matrix and $\mathbf{y}(t-1)$ is the activation of the output node at time $t-1$. In this work feedback connections are not used, so $\mathbf{W}_{res}^{out}$is set to zero. Once trained, the actual output of the network is calculated according to:

$$\hat{\mathbf{y}}(t) = f^{out}(\mathbf{W}_{out}^{inp}\mathbf{u}(t) + \mathbf{W}_{out}^{res}\mathbf{x}(t) + \mathbf{W}_{out}^{bias}) \tag{3.4}$$

where f^{out} is the output neurons' activation function (in this research a linear readout is used, hence the identity function is used), $\mathbf{W}_{out}^{inp}$ is the weight matrix from the input neurons to the output neurons and $\mathbf{W}_{out}^{bias}$ is a bias matrix applied to the output neurons. In this work, input-to-output connections were not used, therefore $\mathbf{W}_{out}^{inp}$ was set to zero. The initialisation, training and testing procedure for a standard ESN is described in the following algorithm.

1. Creation

    - Create the ESN architecture specifying the number of neurons in each layer. Parameters which control the dynamics of the reservoir should also be specified (see section 3.4 below for further details).

2. Initialisation

    - Randomly initialise the input-reservoir, input-output (where present), reservoir-reservoir, reservoir-output, output-output (where present) and output-reservoir weights (where present) and rescale $\mathbf{W}_{res}^{res}$ weights by the *spectral radius*, ρ (more details on the spectral radius are presented later in section 3.4.2 and equation 3.10 while other important network parameters are introduced in section 3.4).

3. Training

    (a) For each training input pattern p:

        i. For each time step, t, in an input pattern:

            - Calculate every reservoir neuron's activation using equation 3.3, multiplying the activations of the input neurons and the activations of the reservoir at time $t-1$ by their corresponding weight matrices; the activation function of the reservoir neuron is then applied to give the activation of the reservoir neuron at time t.

        ii. End of p^{th} input pattern.

    (b) End of all input patterns.

    (c) Once the reservoir activations for every input for the training data have been calculated, calculate the output weights using a linear regression tech-

nique on the reservoir activations and the target output (the Moore-Penrose pseudo-inverse [205, 206], Wiener-Hopf [207] and ridge regression [208] are commonly used). These are the only weights to be changed during training.

4. Testing

   (a) For every test input pattern q

      i. For each time step, t, in a test input pattern:

         - Calculate every reservoir neuron's activation according to equation 3.3, as outlined in the training procedure above.

      ii. End of q^{th} test input pattern.

   (b) End of all input patterns.

   (c) Once reservoir activations have been calculated, calculate the actual output by multiplying the reservoir neuron's activation based on the test data by the reservoir-output weights determined during training. The output activation function is then applied to the result as shown in equation 3.4 .

   (d) Calculate the network error by taking the difference between the target output, $\mathbf{y}_{tgt}$, and the actual output, $\hat{\mathbf{y}}$.

The training approach presented above calculates the activations in batch and, therefore, is known as off-line or batch training. Online training can also be performed using a number of algorithms including the Least Mean Squares (LMS) algorithm [209] and the Recursive Least Squares (RLS) [207] learning algorithm. In the introductory paper on ESNs by Jaeger [64] it was found that the LMS algorithm took too long to converge, while the RLS algorithm offered good performance.

At the heart of the RC training procedure is a memory-less linear regression which is performed using the reservoir activations and the target output values. Any of the regression techniques mentioned in 3(c) above, which are based on OLS, can be used to compute the output weights to satisfy:

$$\mathbf{W}_{out}^{res} = \underset{\mathbf{W}_{out}^{res}}{\operatorname{argmin}} \parallel \mathbf{X} \times \mathbf{W}_{out}^{res} - \mathbf{y}_{tgt} \parallel_2 \tag{3.5}$$

where $\mathbf{X}$ is an $N \times T$ matrix of all reservoir activations, N is the number of neurons and T is the total number of time steps of the dataset, $\parallel ... \parallel_2$ is the Euclidean norm which for a vector A is given by $\sqrt{a_1^2 + \ldots + a_n^2}$, and $\underset{\mathbf{W}_{out}^{res}}{argmin}$ denotes the argument of minimum which is the set of values for $\mathbf{W}_{out}^{res}$ which give the minimal error.

When using the Moore-Penrose pseudo-inverse [205, 206] the output weights are calculated using:

$$\mathbf{W}_{out}^{res} = \mathbf{X}^{\dagger}\mathbf{Y}_{tgt} \tag{3.6}$$

where $\mathbf{X}^{\dagger}$ is the pseudo-inverse of $\mathbf{X}$ which equals $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.

If ridge regression [208] is chosen to train the output weights, it minimises the penalised loss function which is expressed formally as:

$$\mathbf{W}_{out}^{res} = \underset{\mathbf{W}_{out}^{res}}{argmin} \parallel \mathbf{W}_{out}^{res}\mathbf{X} - \mathbf{y}_{tgt} \parallel_2 + \lambda \parallel \mathbf{W}_{out}^{res} \parallel_2 \tag{3.7}$$

where λ is the optimal regularisation parameter (see section 3.3.2.3 below). The solution to equation 3.7 and the calculation of the output weights are found according to:

$$\mathbf{W}_{out}^{res} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}_{tgt} \tag{3.8}$$

where λ (the optimal regularisation parameter) is found in this work using a grid search with cross-validation and $\mathbf{I}$ is the identity matrix. All other values are as before.

3.3.2.1 Alternative reservoir operations

The operation of an ESN so far in this chapter has focused mainly on its use in batch mode (or off-line), i.e. presenting all of the training data to the reservoir and calculating its activations, after which the output weights are calculated. ESNs can be used in several other modes, some of which are briefly outlined in this section.

- Batch (off-line): the reservoir activations are calculated for all of the training data, after which the output weights are calculated as well as the network error. After this process, the network's performance when applied to unseen test data can be evaluated.

- On-line: the reservoir output weights are calculated in real-time using LMS or RLS.

- Free running: the output of the network is fed back into the reservoir as input. This is often used in signal generation tasks where the reservoir is first trained with a teacher signal and then is required to predict the next time step (known as teacher forcing). After training, the reservoir operates in free running mode where its output is presented back as input. This mode of operation can be used to test for stability as any undesirable reservoir dynamics will lead to an error which quickly exponentiates. This is the mode of operation that was used by Jaeger when applying a reservoir to the Mackey-Glass dataset [64].

3.3.2.2 Cross-validation

Cross-validation works by splitting a dataset into N samples (or folds as they are often called, hence the term N-fold cross-validation). During training, $N-1$ samples are used, with the remaining sample used as the test data. This process is repeated N times where the sample used as the test data is altered each time. Cross-validation helps to avoid misleading results that can be obtained through, often accidental, poor choice of training and testing sets which can be misrepresentative of the dataset as a whole, resulting in reduced performance.

When finding the optimal regularisation parameter using a grid search, cross-validation uses three subsets of the dataset, the training set, validation set and test set. During this procedure, the training set and validation set are used to evaluate a network's performance using different values of the regularisation parameter which gives an error when the network is applied to the validation set. Once this optimal parameter is found, the performance of the network is evaluated using the test subset. The subsets are then shuffled and the process is repeated. While this may be computationally expensive with large datasets and a large number of folds, it does improve performance and ensures that regularisation is near optimal [210].

3.3.2.3 Regularisation

Using ridge regression to train the output weights has the advantage of regularising the reservoir readout during training, which finds the optimal trade-off between model complexity and its generalisation abilities, avoiding overfitting while giving good performance on unseen data. An example of this was shown by Wyffels and Schrauwen [211] where network errors increased when trained without ridge regression as the number of

neurons increased when compared to reservoirs of the same size which were regularised using ridge regression. Regularisation leads to a network which is more robust to noise present in unseen data, resulting in better performance. Regularisation can also be achieved by adding noise to the input data (see section 3.4.8 below for more details).

3.3.2.4 Reservoir adaptation

Adapting the reservoir either prior to training or during training has recently become an interesting area of RC where the reservoir activations are adapted based on either the input signal only (i.e. in an unsupervised manner using no target output) or both the input and output signals (i.e. in a supervised manner).

Intrinsic plasticity [212] is perhaps the most well-known unsupervised reservoir adaptation technique which has been observed in the visual cortex [213], for example. IP has been successfully applied to reservoir based techniques by a variety of research groups for the BPDC algorithm [214, 215], as well as ESNs [216, 217]. This is achieved by tuning the neurons to have an activation value which approximates a distribution, such as an exponential distribution, which maximises their entropy. One study by Verstraeten et al [217] applied IP to ESNs in order to gain Gaussian distributions using the hyperbolic tangent activation function where it was found that an IP-trained reservoir's variability in performance was reduced when tuning certain parameters, such as the spectral radius and the variance of the output distribution. This meant that reservoir settings could be evaluated accurately without the need to create many random reservoirs. A recent novel architecture named *Simple Cycle Reservoir* [218] also overcomes this problem by using a deterministic approach to input and reservoir weight creation which will be covered in more detail in section 3.6.4 below.

Other adaptation techniques include the use of a gradient descent approach to find the smallest network error [219] and the use of an evolutionary approach to tune the reservoir parameters [220, 221]. For the work reported here, no reservoir pre-adaptation was performed as the main focus of this research was the amount of non-linearity and memory a reservoir could contain and hence, was not relevant to this research.

3.4 Tunable reservoir parameters and reservoir dynamics

As the section 3.3.2 above showed, training of ESNs is very simple. In fact, upon first inspection it may appear counter-intuitive that the simplicity of training only the output weights of a sparse and randomly connected reservoir can create a network which can give an output close to the target output, especially when applied to unseen test data. However, work by Schiller and Steil [222] investigated two training algorithms of RANNs (more specifically, the Atiya-Parlos learning algorithm [204] and real-time recurrent learning [223]) and found that during training the majority of the weight changes were between the hidden and output layers; therefore training only these weights using a quick and optimal linear regression technique makes sense. Under certain conditions, RANNs have been shown to be universal approximators of dynamical systems [60, 61, 62]. Since then, Maass et al [224, 225] have shown that reservoirs can be made robust to noise through their training, making them computationally universal, i.e. the reservoir can approximate any given real-time computation on real-time systems with time-varying inputs and a fading memory.

As with all machine learning techniques, however, there are certain parameters to optimise in order to obtain a reservoir which offers good performance in a specific domain.

This section outlines the main parameters of a reservoir which require optimisation. Currently, the common approach to reservoir optimisation involves random selection of values and a broad empirical sweep to find the parameters which enable optimal performance for a given dataset [200, 217, 219, 226] although other optimisation approaches have been recently proposed, some of which such as IP and evolutionary approaches are outlined above in section 3.3.2.4 and in Jaeger et al [219] where a stochastic gradient descent based approach to find the optimal parameters was investigated. This section introduces the reservoir parameters which can be varied to improve performance, along with the effects of altering the parameters on the reservoir dynamics (certain desirable reservoir dynamics are then discussed in more detail in section 3.5).

As reservoirs are a RANN, their most common application is the generation, prediction or classification of data from time-series domains. In order to do this, and to satisfy the *echo state property* (see section 3.5.1 below for more details on the echo state property), a reservoir needs to have a fading short term memory. As well as memory, the reservoir must be able to map the input data, which is often non-linearly separable, onto a high dimensional state space, in which hyperplanes can be drawn by the readout layer between data-points belonging to different classes, in a classification task for example. This allows the reservoir to classify non-linearly separable data. For most real-world tasks, especially those whose data is highly non-linear and time-series based, a combination of memory and non-linear transformation capabilities is required, but, as this section and section 3.7 below will show, there is a trade-off between the amount of these two capabilities that a reservoir can possess. Figure 3.3 shows a typical sigmoid activation function curve, where the slope of the gain is largest when a neuron's working point is around the origin, which is also the linear part of the function. Although a less dynamical reservoir (i.e. one whose memory fades after only

a short period of time or is unstable) is undesirable as it often leads to a decrease in its performance, for datasets which require complex highly non-linear mapping capabilities, some of the reservoir neuron's working points must move towards the non-linear regions of their activation functions. The effect of changing network parameters on the movement of the reservoir neuron's operating point and the subsequent effect on the dynamics of the reservoir is outlined in the remainder of this section.

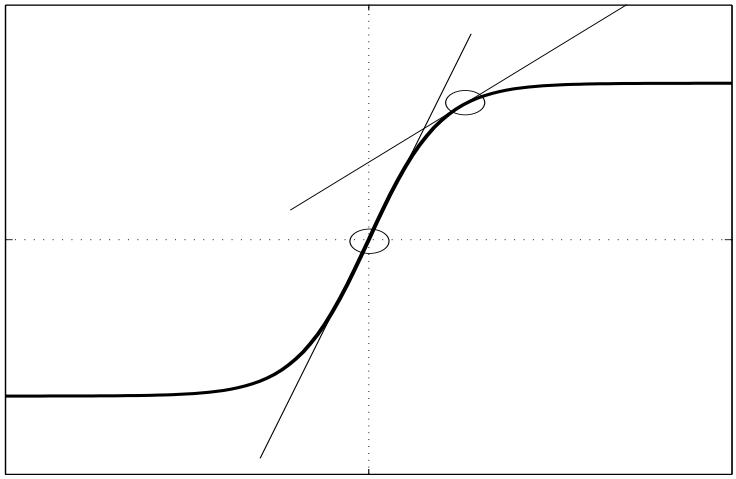


Figure 3.3: A sigmoid activation function showing two different gains which are determined by the slope of the corresponding tangents. When driven by input, the working point of the neuron moves away from the origin where its gain is the largest to a more non-linear part of the function where its gain decreases. In the saturated regions (i.e. the regions towards the extremes) of the activation function the neurons lose their power to discriminate between different inputs, which is indicated by a reduction in the neuron's gain.

3.4.1 Input scaling

The input scaling (ι) determines the extent to which the reservoir neuron activations are driven by the input, in other words the extent to which the current input impacts the next state of the reservoir [227]. The default value is usually unity, but changing the input scaling can have a major impact on performance. When changing ι the initial input weight matrix, $\mathbf{W}_{res}^{inp}$, is updated according to:

$$\mathbf{W}_{res}^{inp'} = \iota \cdot \mathbf{W}_{res}^{inp} \tag{3.9}$$

where $\mathbf{W}_{res}^{inp}$ in this research contains values of either +0.1 or -0.1 chosen uniformly.

Figure 3.4 shows the effect of changing the input scaling on the activation of a reservoir neuron when fed with random input data. A small input scale usually creates a linear reservoir (assuming that the other parameters, such as the spectral radius, have non-extremal values) as most of the working points of the reservoir neurons stay within the linear range of their activation functions, as shown by the second plot in Figure 3.4 below. On the other hand, larger input scales move the working point of the reservoir neurons towards the non-linear parts of their activations. A very large input scale leads to the saturation of the entire reservoir, where all of its neurons' activations are close to either -1 or +1 and remain at this saturation point or alternate between extremes for the majority of the dataset (as shown in the bottom plot of Figure 3.4). In this situation, the gain of the activation function is at its smallest, as it has moved either upwards or downwards along the activation function and the neurons of the reservoir may no longer be able to discriminate between small changes in the input data.

For highly non-linear data, whose values change drastically from one time step to the next, and which requires very little in terms of short-term memory a high input scale

to a reservoir may offer best performance as it is required to map the input data from one region of its high dimensional state space to another region in order to classify the data correctly which is aided with a larger state space (this is later confirmed for highly non-linear data which requires little short-term memory and is shown in Chapter 6). While a saturated reservoir may offer the best performance, it may also be unstable as a result of its neuron activations shifting from one extreme to another.

Figure 3.5 is a schematic representation of the effect of changing the input scale on the *tanh* activation function, where high values of the input scale increase the range of the summed input, S, to the neuron (X axis) where $S = (\mathbf{W}_{res}^{inp}\mathbf{u}(t-1) + \mathbf{W}_{res}^{res}\mathbf{x}(t-1) + \mathbf{W}_{res}^{out}\mathbf{y}(t-1) + \mathbf{W}_{res}^{bias})$. By increasing the input scale, the summed input of a neuron increases, as indicated by S'.

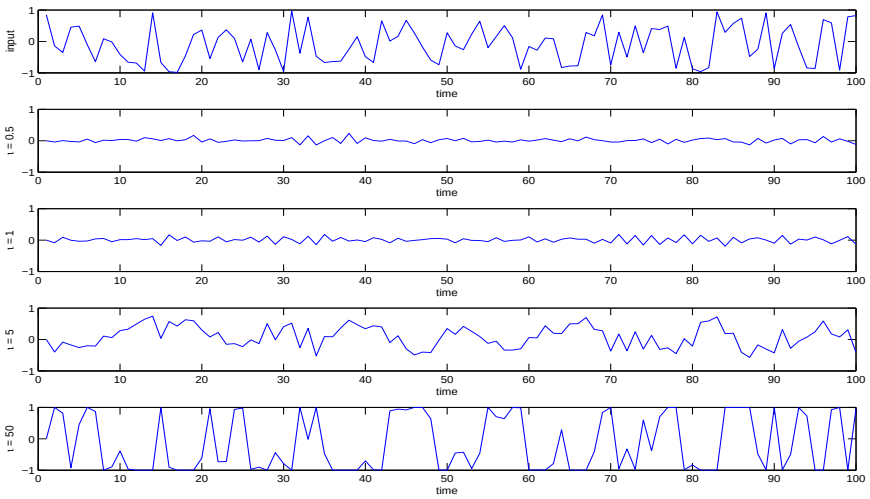


Figure 3.4: The effect of input scaling on the activations of a reservoir neuron. The top plot shows the input, while the second, third and fourth plots show the activations of the reservoir neurons with input scales of 0.5, 1, 5 and 50, respectively. Typically an input scale of 1 is used, but values below and above can improve performance for certain tasks. Extreme values of input scale, such as is the case in the bottom plot lead to the saturation of the neuron, where its activation either stays at its extreme values or alternates between the two for the majority of time steps. In this case, the generalisation capabilities of the neuron are impeded as it is less able to discriminate between its inputs. As the same input scale is applied to all reservoir neurons, the above also applies to the entire reservoir population.

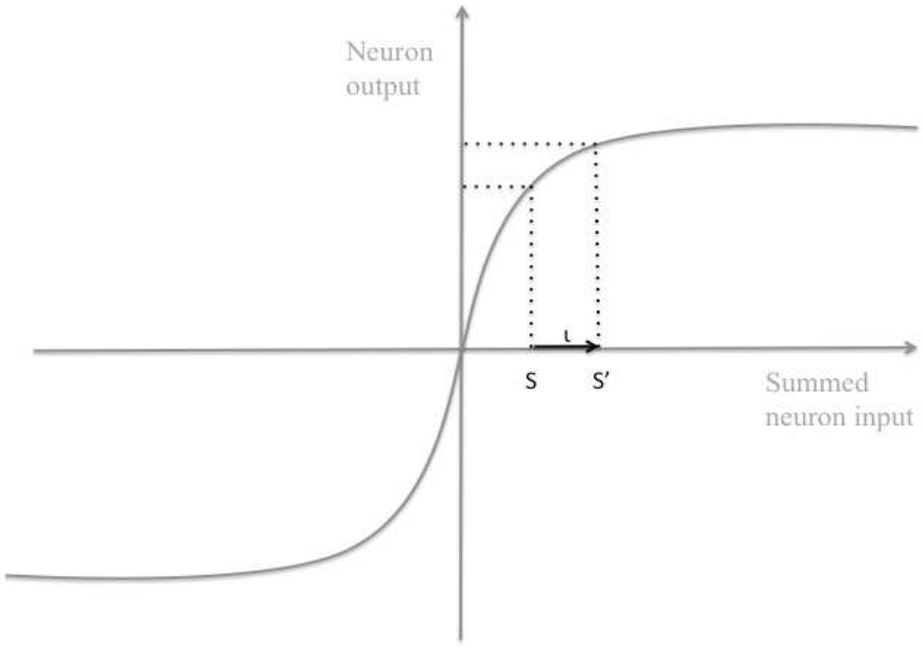


Figure 3.5: A schematic representation of the effect of increasing the input scale on the activation function of a neuron. Larger values of input scale increase the range of the neuron's summed input (X axis) from the original summed input, S, to the summed input, S'. Very large values move the working point of the neuron to its saturated regions.

3.4.2 Spectral radius

The spectral radius (ρ) is the absolute value of the highest eigenvalue of the reservoir weight matrix which is used to rescale the weights of $\mathbf{W}_{res}^{res}$ in order to satisfy the echo state property. The echo state property essentially allows the reservoir to possess fading memory, or "echoes", of its past inputs (the echo state property is discussed in more detail in section 3.5.1 below). After reservoir weight initialisation, the weights of the reservoir are rescaled by the spectral radius:

$$\mathbf{W}_{res}^{res'} = \frac{\rho.\mathbf{W}_{res}^{res}}{\lambda_{max}} \tag{3.10}$$

where λ_{max} is the maximum absolute eigenvalue of the initialised reservoir weight matrix, $\mathbf{W}_{res}^{res}$, whose weights are drawn randomly from a Gaussian distribution with zero mean and a variance of 1.

Varying the spectral radius affects the amount of memory a reservoir contains: values much smaller than unity decrease the memory capacity of the reservoir as the reservoir neurons' activations become very contractive around their origin which means that any activation activity quickly dies out after a period of no input. Larger values close to unity increase the memory capacity as the neurons remain active for longer after no input [227]. As will be discussed below in section 3.5.1 a spectral radius value above unity does not necessarily result in a violation of the echo state property, although values larger than unity do reduce the memory capacity of the network and can lead to an unstable reservoir which can give decreased performance.

Varying the spectral radius has a similar effect on the dynamics of the reservoir as the input scaling: values around unity generally create a linear reservoir with a high memory capacity, whereas values larger than unity create a very non-linear reservoir which in turn has a low memory capacity. Where the impact of the two parameters differs is when using small values: a small spectral radius value decreases the memory capacity of the reservoir as its activations quickly die out after no input, whereas small values of the input scale create a reservoir which operates mainly in the linear regime and as a result has a higher memory capacity. Extremely high values of spectral radius or input scaling lead to the saturation of the reservoir neurons' activations, reducing

the generalisation capabilities of the reservoir. However, the effect of changing the spectral radius is almost removed when a high input scale is present as was shown by Dutoit [227].

Although varying the spectral radius and the input scale can have a similar effect on the reservoir, the mechanism by which the two parameters influence the current state of the reservoir differ: the input scale influences the effect of the previous input on the current state of the reservoir, while the spectral radius influences the effect of the previous reservoir state (which has been influenced by all previous inputs) on the current reservoir state [227]. Figure 3.6 shows the effect of varying the spectral radius on the reservoir dynamics and reservoir neurons' activation distributions (right hand plots). In the top graph a small value of spectral radius creates a reservoir which is too stable as indicated in the distribution plot where most of the activations are centred around zero, which is also the linear part of the activation. The centre plot shows a reservoir whose dynamics are most desirable, with the majority of its neurons' activations around the linear region but with some non-linear neurons also. The bottom plot shows a reservoir with a spectral radius higher than unity which is reflected in the distribution of the reservoir neuron's activations, as most are saturated at the extremes of their non-linearities.

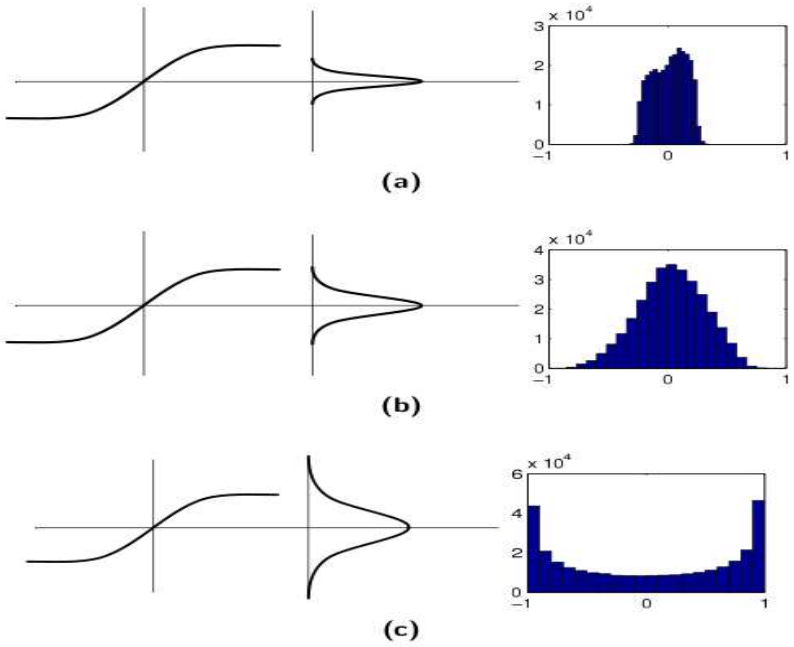


Figure 3.6: The effect of the spectral radius on the distribution of the reservoir neurons' activations (reproduced from Verstraeten [210]). The left hand column shows the sigmoid activation function, while the middle column shows the spread of the activation function output and the right column shows the explicit reservoir neuron activation values. The top plot (a) shows the effect of a small spectral radius which gives a reservoir with linear neurons as most activations are centred around zero as shown by the right hand column. The middle plot (b) shows the ideal situation, where the reservoir neurons are spread over non-linear and linear parts of their activations. The bottom plot (c) shows a reservoir with a spectral radius higher than unity where the majority of neuron's activations are saturated at their extremes which reduces their ability to discriminate between inputs.

Increasing the spectral radius has the same effect as the input scale on the summed input of a neuron, the larger the spectral radius, the larger the summed input, and hence, the reader is referred to Figure 3.5 for a schematic representation of this effect.

3.4.3 Leak rate

The leak rate (δ) determines the extent to which a neuron's activation decreases over a period of time. This was introduced originally with the aim of altering the dynamics of the reservoir to match the time-scale (i.e. the rate of change of a signal) of a dataset, for example a very slow sinewave. Using standard *tanh* neurons, such slow changing data is difficult to model as the neurons are influenced only indirectly by their previous value and have no actual memory. This idea was originally introduced by Jaeger [64] as part of the activation function of a leaky integrator neuron which through the use of a leaking decay rate had some internal memory. It was shown that matching the speed of the internal dynamics of the reservoir with the time-scale of some datasets can result in improved performance [228].

The leak rate can be applied to a neuron either before the non-linearity, which is often used in continuous time RANNs such as the BPDC algorithm [181], after the non-linearity, as introduced by Jaeger [64], and over the non-linearity as introduced by Schrauwen et al [228]. In this work, the leak rate is applied over the non-linearity as it stabilises the neuron through the constraint of the non-linearity of the activation function and, as a result, has been shown to give improved performance when compared to the two other approaches in solving the speech recognition task which is outlined in Chapter 5.1.4 below. Neurons which contain a leak rate placed over the non-linearity are updated according to [228]:

$$\mathbf{x}(t) = f\left((1-\delta)\mathbf{x}(t-1) + \delta(\mathbf{W}_{res}^{inp}\mathbf{u}(t-1) + \mathbf{W}_{res}^{res}\mathbf{x}(t-1)\right) \tag{3.11}$$

Decreasing the leak rate increases the ability of reservoir neurons to recall inputs presented further in the past, but decreases their ability to recall the most recent

inputs, whereas a large leak rate (note that the largest leak rate cannot be larger than unity) has the opposite effect where the reservoir is able to remember more recent inputs than ones presented further in the past [210]. Therefore the optimal leak rate depends on the memory requirements of the task at hand.

3.4.4 Sampling of the dataset

As mentioned above regarding the leak rate, it was found that a best performing reservoir often has the same, or very similar, timescale present within its internal dynamics as that present in the dataset [228]. Resampling of the dataset is another way of achieving this tuning. Upsampling involves increasing the time that each datapoint exists within the dataset, effectively slowing the dataset down. Downsampling has the opposite effect, in that it summarises datapoints drawn from the dataset over a given period, speeding up the rate of change within the dataset which can be useful for slow changing and/or very large datasets. Depending on the characteristics of the dataset, up or down-sampling the data can have an impact on the performance of the reservoir.

3.4.5 Applying bias to reservoir neurons

The bias can be applied to all reservoir neurons through the matrix $\mathbf{W}_{res}^{bias}$ which is sampled from a normal distribution and multiplied by the bias scalar (β) which controls the amount of bias fed into the reservoir. The use of bias to the reservoir can help to move the reservoir neuron activations away from their origin when applying the network to generative tasks (i.e. free-running as was introduced in section 3.3.2.1 above). Bias can also be used to stabilise the network by using a large value to move the reservoir neuron activations towards the non-linear part of their activation function, which can be useful with highly non-linear datasets [182, 227]. Applying bias to the

readout can also aid performance when the target output of a dataset has a mean which is not equal to zero [182]. Figure 3.7 is a schematic representation of the impact of changing the bias on the activation function of a neuron.

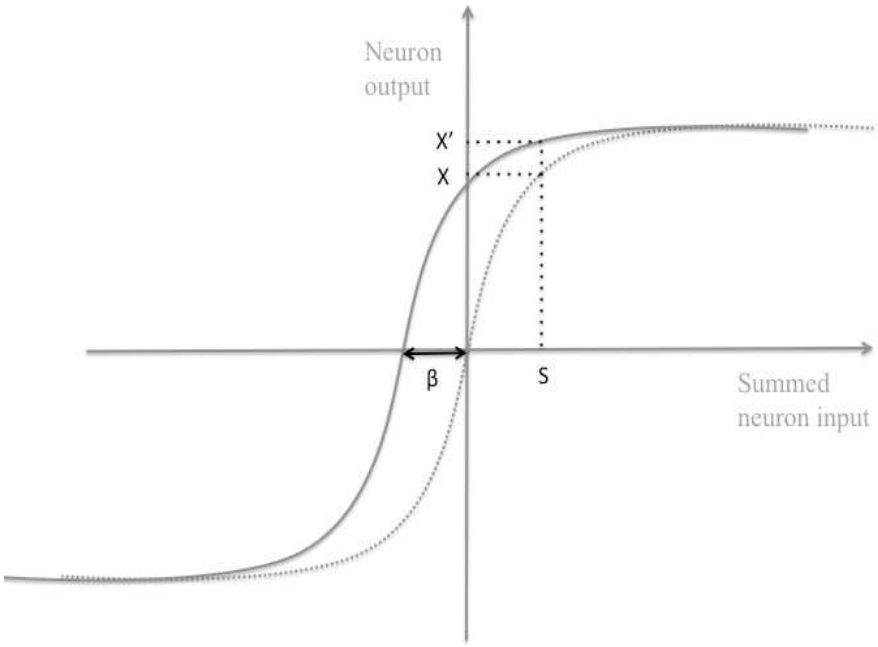


Figure 3.7: A schematic representation of the effect of applying a bias to a reservoir neuron, which shifts the origin of the *tanh* activation function along the X axis where the original *tanh* function is shown as the dashed line, while the new *tanh* function is shown as the solid line. As a result of using a positive bias scalar (β), the new output of the neuron denoted X' is increased.

The effect of changing the bias on the dynamics of the reservoir has also been studied by Verstraeten et al [226] who showed that large values of bias can increase the amount of asymmetric non-linearity a neuron possesses by applying a shift relative to its origin. Typically, as a result of the *tanh* activation function with zero-mean input, the non-

linearity of a neuron is symmetric. In an extended version of the XOR task (see section 5.1.3 in Chapter 5 below for more details on this task) it was shown that, with small values of input scale, changing the bias to give the reservoir some asymmetric non-linearity mapping capabilities improved the performance of a reservoir, although a trade-off was present as the symmetric non-linearities of the task increased.

3.4.6 Reservoir size

Reservoir size is also a parameter to consider tuning to achieve improved performance. In an investigative study by Verstraeten et al [200], it was found that as the reservoir size increased, so did the memory capacity of the network. For linear neurons, it has been theoretically proven (Jaeger [229]) that the maximal memory capacity of a reservoir is equal to its number of neurons. It was also found that, for tasks such as the memory capacity (MC) task (see section 3.6.4 below) where longer memory is required from the network, the increase in the size of the reservoir improved performance as more neurons resulted in the ability to recall inputs from further in the past. For the well studied NARMA task, larger reservoirs led to a decrease in the variability in performance of the reservoir as the readout layer was able to extract the information required by the task from the reservoir dynamics (in the case of small reservoirs, the proportion of neurons which perform poorly was much higher) [200].

3.4.7 Activation functions

Using different activation functions can have an impact on the performance of a reservoir. The activation function used has a big influence on the memory capacity a neuron can possess. For linear neurons it has already been stated that they have a maximal memory capacity that is equal to the number of neurons inside the reservoir.

Although linear neurons have greater memory capacity than neurons with sigmoid activation functions, they struggle with highly non-linearly separable data. Schrauwen et al [216] investigated reservoirs which were adapted using IP. In one experiment using the MC task (see section 3.6.4 for more details on this dataset), it was found that sigmoid neurons had a larger memory capacity than neurons with a Fermi activation function. Verstraeten et al [200] studied the performance of different activation functions on different tasks. For a spoken digit recognition task (see section 5.1.4 of Chapter 5 for more details), linear neurons were found to perform worst with the highest error rate, while sigmoid neurons offered slightly better performance.

3.4.8 Noise insertion

Adding noise to the dataset can improve performance for some problem domains by regularising the reservoir and aiding in the avoidance of over-fitting. The addition of noise usually results in an increase in the training error, but a decrease in test error. It was reported by Wyffels et al [230] that noise insertion achieved similar performance to ridge regression as it regularised the reservoir against overfitting, but the random nature of noise made it difficult to reproduce results reliably for each reservoir architecture. Adding noise also becomes problematic if using output-to-reservoir feedback connections, as the activations of the reservoir need to be calculated for each level of noise which can increase the computation requirements [210].

3.5 Desirable reservoir dynamics and properties

The previous section outlined the parameters of a reservoir which can be varied to improve performance. This section gives an overview of the desirable dynamics of

a reservoir one should obtain in order to give good performance when applied to a problem domain.

3.5.1 The Echo State Property

In order to obtain a reservoir which performs well on a given task, Jaeger [64] outlined that an echo state type reservoir should satisfy the echo state property. This essentially means that the reservoir should have a fading memory, where it remembers just enough about previous inputs such that this memory does not disturb the reservoir's representation of the current input. It also means that the current inputs have a greater influence on the current state of the reservoir neurons than inputs presented in the past and that any inputs should eventually fade away in the reservoir after a period of time, allowing the reservoir to settle back to a stable state. A reservoir which satisfies the echo state property should, for any input from a compact set, wash out (remove) any initial reservoir conditions created at initialisation which impact the reservoir activations at a rate which is independent from the input [219].

In order to satisfy the echo state property it is recommended that the largest absolute value of the eigenvalues of the reservoir weight matrix, known as the spectral radius (ρ) (as introduced in section 3.4.2), should be close to, but not equal to unity [64], making the reservoir weight matrix contractive. The spectral radius can be interpreted as the forgetting factor of the network, as it determines the extent to which the network forgets its previous inputs. Any values above unity and the reservoir becomes asymptotically unstable at the zero state and the echo state property disappears for zero-valued input, although a value higher than unity for non-zero data does not result in the disappearance of the echo state property as has often been misinterpreted [63, 200, 231].

When using clusters of small reservoirs, Jarvis et al [232] recently showed that larger values of ρ were possible without creating unstable reservoirs and where clusters became unstable they were less likely to make the whole network unstable as a result of sparse connections between clusters. As the spectral radius is influenced only by the weights within the reservoir and does not take into account any inputs presented to the network, a value higher than unity after input presentation does not necessarily mean that the echo state property has been violated. This has been shown in previous studies, for example see Verstraeten et al [233] for experimental proof. As a general guide, Jaeger [64] recommended using values close to unity for tasks which require long memory and a smaller value where less memory is required.

In the same introductory paper, Jaeger [64] also stated the sufficient condition for the echo state property to exist for any input, including zero-valued inputs, that largest singular value of the reservoir weight matrix, $\mathbf{W}_{res}^{res}$ should be smaller than unity. This is only a guideline however, as just fulfilling this criterion will not always lead to the best performing reservoir.

The echo state condition has since been revised by Buehner and Young [234] who presented a tighter bound for the echo state property which is based on a less restrictive test for global asymptotic stability of the reservoir and ensures that echo states exist for all inputs. This tighter bound allows for a more rigorous analysis of the reservoir to assess the presence of the echo state property as was shown in their study, but can be computationally expensive as reported in Verstraeten [210] where, with a 500 neuron reservoir, the tighter bound for the spectral radius took 21 seconds to compute, compared to 0.6 seconds for the standard spectral radius. While the increase in computation is not huge, coupled with a larger network and the intensive grid search training

algorithm, the computational increase in other scenarios could be much higher.

When a network is presented with input the *effective* spectral radius (ρ_{eff}) [235] of its reservoir decreases [210]. The effective spectral radius takes into account the input to the reservoir via the activations of the reservoir neurons, which differs from the standard spectral radius which only takes the initial reservoir weights into account. These weights never change once the weights have been globally rescaled by the spectral radius shown earlier in equation 3.10; hence, the spectral radius is often called a static measure of the reservoir dynamics [210]. The fact that the spectral radius only takes the reservoir weights into account offers an explanation as to why a spectral radius value larger than unity can still produce a reservoir with the echo state property when driven by a sufficiently long sequence of input data. The use of the spectral radius as a performance indicator when using activation functions that are non-linear around their origin also becomes less reliable as it only really applies to activation functions which are linear around their origin [210], such as *tanh* neurons. The reservoir's neurons working point moves either up or down towards a more non-linear region of the activation function as a result of being presented with input data, and as a result the gain (the slope of the tangent) becomes smaller, giving a less dynamical reservoir (which decreases ρ_{eff}).

3.5.2 Computation on the edge of stability

An area of ongoing research within RC is determining what characteristics a reservoir must possess in order to enable it to perform well in a given problem domain. As well as fulfilling the echo state property which enables the reservoir to have a fading memory, the reservoir must contain neurons which map the input onto a rich and dynamic state space and behave somewhere between a too stable and a unstable regime.

Where the reservoir is too stable, its activations stay in the same regime where the presented input has little effect on the reservoir. In an unstable regime, the activations of the reservoir neurons change exponentially from one input pattern to the next, resulting in a highly sensitive reservoir which contains little information regarding the inputs as it never settles towards one (or more) equilibrium point(s), also known as attractors, in its state space. As the activations of an unstable reservoir often deviate exponentially from its input, the echo state property is violated as any slight change in the data or initial starting conditions are also exponentially changed. In essence, a reservoir which is either too stable or unstable will often perform worse than one which contains both characteristics as it is unable to give a good representation of the input data in its internal state space, which reduces its generalisation capabilities. Reservoirs which operate on the edge of stability have been shown to be computationally more powerful than those which are too stable or too unstable in the work by Bertschinger and Natschläger [236]. This has been proposed for both LSMs [237, 238] and ESNs [182].

3.6 Interpreting reservoir dynamics

The search for appropriate and accurate performance measures of reservoirs has become an area of ongoing active research. Although reservoirs have been shown to perform well in particular tasks it is unclear what characteristics of the network and/or problem domains enable them to perform so well. Therefore, developing informative and accurate performance measures of reservoirs is an important research area if rules for creating and using reservoir techniques given particular kinds of tasks are to be obtained.

As mentioned earlier, the spectral radius is not always an indication of good performance capabilities as it is only a good measure of a reservoir's dynamics when using networks with no input [235] as a value larger than unity in this case leads to the violation of the echo state property. Indeed, when a reservoir is applied to data, some studies have found that a value greater than unity results in a better performing network [200, 231]. The spectral radius has also been found to be very dependent on the bias or scaling of the input for linear systems [239]. Ozturk et al [235] showed that with the same spectral radius, different ESNs gave varying error rates for the same task, where mean squared errors between 10^4 to 10^8 were observed. This is due to the weight randomisation at network creation and shows that the spectral radius is a poor indicator of performance.

3.6.1 Lyapunov exponents

As a result of the unsuitability of using the spectral radius as a performance measure, work carried out by Verstraeten and Schrauwen [200, 240] has led to the use of the Lyapunov exponents (LEs) [241] of a reservoir as a performance measure. This measure estimates the excitability of a dynamic system (in this work, a reservoir) around a certain point within its state space [240]. When a system is disturbed by an infinite input sequence, a stable dynamical system will eventually settle towards one (or more) attractor(s) giving a negative Lyapunov exponent; while in a chaotic system, the reservoir will shift exponentially away from its trajectory in one or more directions, never settling towards an attractor in its state space. Here trajectory comes from the field of control theory and is defined as the states of the dynamic system, i.e. the reservoir, over a period of time. In a chaotic system, a Lyapunov exponent will be larger than zero [210]. This measure takes into account both the reservoir dynamics and the cur-

rent task-dependent input, and can indicate an optimal value for the global scaling of the weight matrix.

LEs were originally applied to spiking neurons by Legenstein and Maass [237] by calculating the Euclidean distance between reservoir activations when the network was presented with a time-shifted input spike over 0.5 ms. This was adapted by Verstraeten et al [200] for sigmoid neurons fed with noisy input (in this case randomly chosen from a normal distribution between -0.8 and +0.8). In this work it was found that using the estimated maximum local Lyapunov exponent ($\widetilde{LLE}_{max}$) gave consistent improved reservoir performance. Later work by Verstraeten and Schrauwen [240] further investigated the use of Lyapunov exponents as a reservoir measure using the Mackey-Glass the 30^{th} order NARMA datasets. This study analysed the estimated local Lyapunov exponents and the minimal singular value (SV) of the Jacobian over the datasets where it was found to offer insight into the dynamics of the reservoir as its value changed when altering the spectral radius and input scale: improved reservoir performance was found when the minimal SV was highest. Maximising the minimal SV does however, result in a trade-off between the excitability and non-linear separation capabilities of the reservoir and the number of degrees of freedom in its state space [240].

The local Lyapunov exponents (LLEs) are temporally local approximations of LEs which are computed using the Jacobian, $\mathbf{J}_f$, of the map of the reservoir activations at each time step $\mathbf{x}(t)$:

$$\mathbf{J}_f(\mathbf{x}(t)) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}(t)) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}(t)) \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}(t)) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}(t)) \end{pmatrix} \tag{3.12}$$

where $\mathbf{x}(t)$ is the vector of activations of the reservoir neurons at time t. All other symbols are as before. Using the *tanh* activation function, the reservoir can be described

by the following map [210]:

$$\mathbf{f}(\mathbf{x}(t+1)) = tanh(\mathbf{x}(t) * \mathbf{W}_{res}^{res}) = \left[tanh(\sum_{i=1}^{n} w_{1i}x_1) \cdots tanh(\sum_{i=1}^{n} w_{ni}x_n) \right] \tag{3.13}$$

for which the Jacobian can be obtained:

$$\begin{aligned} \mathbf{J}_f(\mathbf{x}(t)) &= \begin{bmatrix} [1 - tanh^2(\sum_{i=1}^{n} w_{1i}x_1)]w_{11} & \cdots & [1 - tanh^2(\sum_{i=1}^{n} w_{1i}x)]w_{1n} \\ \vdots & & \vdots \\ [1 - tanh^2(\sum_{i=1}^{n} w_{ni}x_n)]\, w_{n1} & \cdots & [1 - tanh^2(\sum_{i=1}^{n} w_{ni}x_n)]w_{nn} \end{bmatrix} \\ &= \begin{bmatrix} [1 - x_1^2(t)]w_{11} & \cdots & [1 - x_1^2(t)]w_{1n} \\ \vdots & & \vdots \\ [1 - x_n^2(t)]\, w_{n1} & \cdots & [1 - x_n^2(t)]w_{nn} \end{bmatrix} \end{aligned} \tag{3.14}$$

which simplifies to:

$$\mathbf{J}_f(\mathbf{x}(t)) = diag\left[1 - x_1^2(t), 1 - x_2^2(t), \cdots, 1 - x_n^2(t)\right] \times \mathbf{W}_{res}^{res} \tag{3.15}$$

The estimated maximal LLE, $\widetilde{LLE}_{max}$, can then be found by:

$$\widetilde{LLE}_{max} = log\left(\max_p \prod_{t=1}^{T} (r_p^t)^{1/t} \right) \tag{3.16}$$

where $r_p^t = \sqrt{|\lambda_p|}$, t is the time step and λ_p is the p^{th} eigenvalue of $J_t J_t^T$.

An $\widetilde{LLE}_{max}$ value greater than zero is a good indicator that the reservoir is unstable as it will deviate exponentially, at least in one direction, away from its trajectory. This exponential deviation from its trajectory results in a poor mapping of the input signal, reducing the reservoir's generalisation capabilities significantly. Lyapunov exponents and their underlying principles are applicable for autonomous systems which run for

an infinite period of time. This rule of instability becomes less clear when applied to reservoirs which are constantly driven by an input of a finite length and as a result have an ever-changing state. As a result a value larger than zero does not always indicate instability.

3.6.2 Average state entropy

Shannon's work on information theory [154] defined entropy as the measure of the amount of information contained within a given signal. The Average State Entropy (ASE) gives a measure of the reservoir activation distribution amplitudes [235]. In other words, it is a measurement of the reservoir's information content averaged over time, the more diverse the responses (or activations) of the reservoir, the higher the ASE value. ASE is based on Renyi's quadratic entropy [242] (as given by Ozturk et al [235]) and is calculated as follows:

$$H_2(X) = -log\left[\frac{1}{N^2}\sum_{j}^{N}\left(\sum_{i}^{N} K_\delta(x_j - x_i\right)\right] \tag{3.17}$$

where K_δ is a Gaussian kernel function, δ is the kernel size (set at 0.3 of the standard deviation of the reservoir activations as in Ozturk et al [235]), N is the number of time steps of the data and x_i is the i^{th} activation of the reservoir neurons. In order to gain optimal reservoirs, the ASE should be maximised as a high value indicates that the reservoir neurons are as diverse and uncorrelated as possible and, as a result, are able to map the inputs onto a high dimensional state space. Maximising the ASE will not always lead to a best performing reservoir however, due to the independence between the target output and the initial reservoir states and the fact that some tasks require a small spectral radius which will result in a smaller ASE value.

3.6.3 Deviation from linearity

The deviation from linearity (δ_ϕ), recently presented by Verstraeten et al [226] is based on the observation that a purely linear reservoir contains energy only at the frequency of the input signal, whereas a reservoir with non-linear characteristics will contain other frequencies. This measure compares the 'energy' ('energy' here is a function of the reservoir's activations) present in a reservoir when stimulated by a periodic input signal (a sine wave) of a particular frequency to the frequency of the input signal. Any other energy in the reservoir at different frequencies is caused by non-linearities present in the reservoir. Calculating δ_ϕ for a given reservoir involves the following steps.

- Create a single input dataset containing 100 sine waves with frequencies linearly spaced from 0.01 to 0.5 Hz (denoted as carrier frequencies in Verstraeten et al [226]).
- Present the reservoir with the first sine wave which has a frequency f_{c1}.
- Perform a Fast Fourier Transform (FFT) of the reservoir activations for all neurons and average for all neurons. Calculate $E_{tot} = mean(FFT(\mathbf{X}) - DC_{cmp}(\mathbf{u})$ where $DC_{cmp}(\mathbf{u})$ is the DC component (the average value) of the input signal and $\mathbf{X}$ is the matrix of all reservoir neurons' activations over the length of the time duration of fc_1.
- Calculate $\delta_\phi = 1 - \frac{E_{c1}}{E_{tot}}$ where E_{c1} is the energy of the FFT at f_{c1}.
- Repeat for all input signals from f_{c2} to f_{c100} and compute the average δ_ϕ.

In this research, the effects of altering the input scale, spectral radius and leak rate on a reservoir's deviation from linearity were investigated. The effect of altering the

input scale and spectral radius of a reservoir on its deviation from linearity are plotted in Figure 3.8.

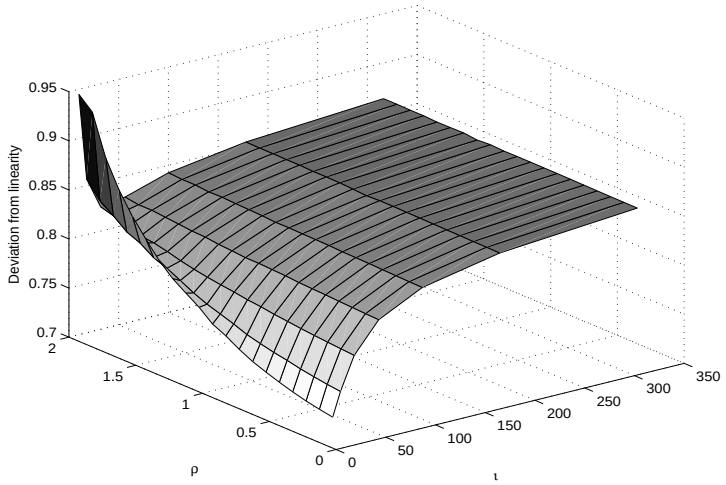


Figure 3.8: The effect of varying the input scale (ι) and spectral radius (ρ) on a reservoir's deviation from linearity. For high values of spectral radius and a low input scale the deviation from linearity of the reservoir is the highest. As the input scale increases, the deviation from linearity values drop and then increase again before levelling out.

The highest value of the deviation from linearity shown in Figure 3.8 is achieved with a small input scale and a large spectral radius which suggests that the spectral radius has a larger influence on the deviation from linearity than the input scale when using a small input scale. The larger influence of the spectral radius on the deviation from linearity measure is also confirmed when increasing the input scale and keeping the spectral radius at lower values where lower amounts of the reservoir's deviation from linearity are observed. Figure 3.8 also shows that generally as the input scale and spectral radius of a reservoir become higher so does its deviation from linearity after an initial fall in deviation from linearity values caused by increasing the input

scale. This general trend is as expected as the working point of the reservoir neurons' activations are moved towards their non-linear regions with higher values of input scale and spectral radius, although the highest levels of non-linearity are achieved with a high spectral radius and a small input scale. The effect of changing the leak rate and the spectral radius on a reservoir's deviation from linearity were investigated and are shown in Figure 3.9.

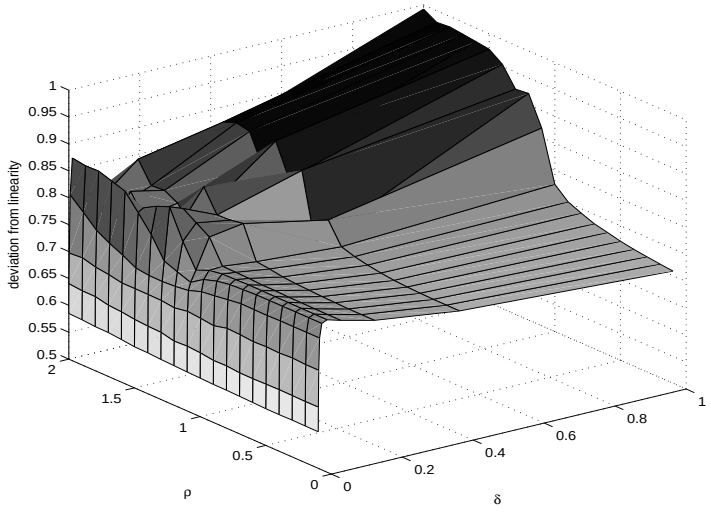


Figure 3.9: The effect of varying the leak rate (δ) and spectral radius (ρ) on a reservoir's deviation from linearity. This shows that as both values increase simultaneously, so do the amount of non-linearities present within the reservoir. For low values of ρ the deviation from linearity tends to decrease as the leak rate increases. With a high leak rate and spectral radius, much higher values for a reservoir's deviation from linearity are observed when compared to high values of input scale and spectral radius as was shown in Figure 3.8 above.

Inspection of Figure 3.9 shows a more complex plot of a reservoir's deviation from linearity when compared to Figure 3.8. The leak rate appears to have a smaller effect on the reservoir's deviation from linearity as shown by the lower curve for spectral radius

values less than unity. For lower values of ρ, smaller values of the leak rate increase a reservoir neuron's memory capacity of inputs presented further in the past, therefore slightly increasing its deviation from linearity as a result the presence of other inputs in the reservoir for last for longer periods of time. When increasing the leak rate values the deviation from linearity values decrease as a result of the reservoir recalling less inputs from previous time steps. As the input scale in this case was kept at unity, the values of the reservoir's deviation from linearity become much higher for high values of spectral radius, which suggests that the input scale restricts the deviation from linearity values somewhat. The effect of altering the input scale and leak rate on a reservoir's deviation from linearity were also investigated as shown in Figure 3.10.

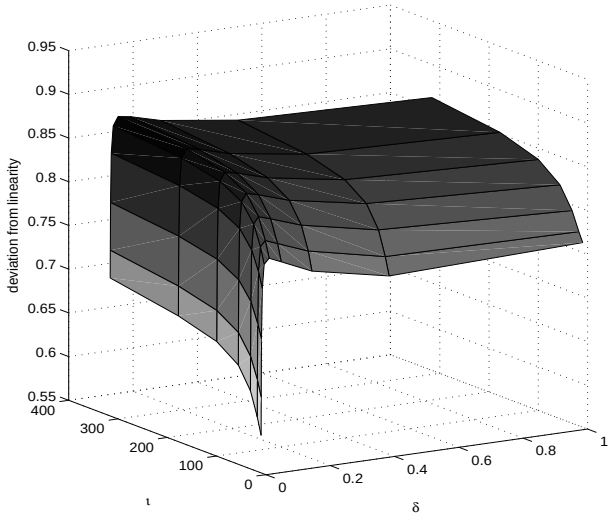


Figure 3.10: The effect of varying the leak rate (δ) and input scale (ι) on a reservoir's deviation from linearity. This shows that higher values are observed for lower values of leak rate and high values of input scale. When compared to Figures 3.8 and 3.9 the deviation from linearity values of the reservoir are lower in this figure which confirms that out of the three parameters, the spectral radius has the most impact on a reservoir's deviation from linearity as the highest deviation from linearity values are observed when ρ is greater than unity.

Figure 3.10 does not show a trend of increasing values of deviation from linearity as the input scale and leak rate increase. Instead, lower values of the leak rate coupled with high values of the input scale give the highest deviation from linearity values. In comparison to Figures 3.8 and 3.9, the deviation from linearity values in Figure 3.10 are smaller, which suggests that the spectral radius has the highest impact on the non-linearities contained in a reservoir as measured by the deviation from linearity (note that in Figure 3.10 ρ was equal to unity). This could be due to the fact that the spectral radius controls the effect of the previous reservoir state at time $t-1$, which

has been influenced by all previous inputs, on the current reservoir state at time t. The input scale affects only the impact of the previous input on the current reservoir state, and therefore the reservoir is only influenced by the input from one time step in the past rather than all of the inputs from the past. By increasing the influence of the previous states the amount of non-linearities present inside a reservoir is also increased as shown by the higher deviation from linearity values when using high spectral radius values.

3.6.4 Memory capacity

So far throughout this chapter the memory of an ESN reservoir has been discussed in an abstract sense. In order to quantify the memory capacity of a reservoir, the Memory Capacity (MC) task was introduced by Jaeger [182] and provides a measure of how well the reservoir can perform the task of remembering past inputs by forcing it to reproduce the input (randomly selected from a uniform distribution between -0.8 and +0.8) which was presented d time steps ago. For example, the 1^{st} output neuron is trained to reproduce the input one timestep ago, the 2^{nd} output neuron two timesteps ago and so on. The activation of each output neuron (d) at time t is, therefore, defined as:

$$\mathbf{y}^d(t) = \mathbf{u}(t-d) \tag{3.18}$$

The memory capacity for the neuron with time delay d, MC_d, is:

$$\mathbf{MC}_d = \frac{Cov^2([\ \hat{\mathbf{y}}^d, \mathbf{y}^d_{tgt}])}{Var[\hat{\mathbf{y}}^d]Var[\mathbf{y}^d_{tgt}]} \tag{3.19}$$

where $\hat{\mathbf{y}}^d$ is the actual output of the network for delay d, $\mathbf{y}^d_{tgt}$ is the target output for delay d and Cov and Var are the covariance and variance. The total memory capacity,

MC for each delay is:

$$\mathbf{MC} = \sum_{d=1}^{\infty} \mathbf{MC}_d \tag{3.20}$$

As previously discussed, the gain of the sigmoid activation function is largest around its origin, which is also the linear part of the activation function where memory capacity is the highest. As the reservoirs move away from the linear region of their activations when driven with external input, their memory capacity decreases. Figure 3.11 shows the effect of varying the input scale and spectral radius on the memory capacity of a typical ESN reservoir.

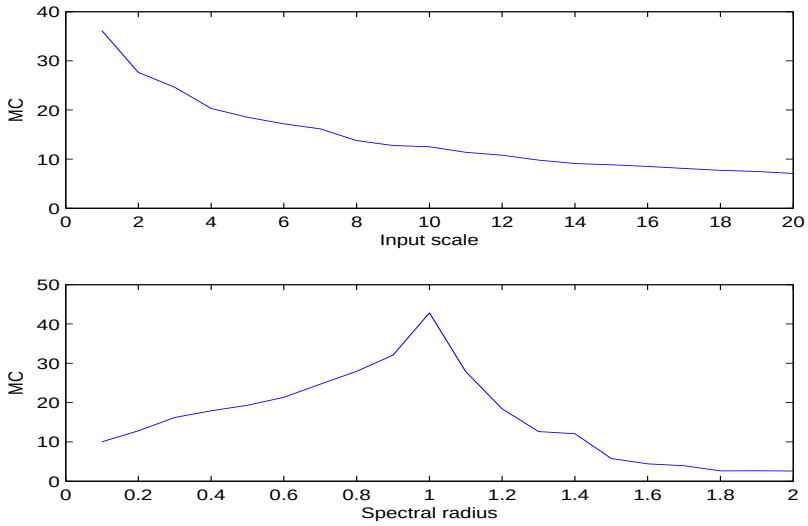


Figure 3.11: The effect of varying the input scale (upper plot) and spectral radius (lower plot) on the memory capacity of a reservoir (in the upper plot the spectral radius is equal to unity, while in the bottom plot the input scale is equal to unity) as defined by the MC measure. Note that as the input scale increases the MC decreases as a result of a more non-linear reservoir, while the maximal MC when varying the spectral radius in this case is for $\rho = 1$: before and after this point the MC decreases as the reservoir becomes more contractive with small values and more non-linear (and eventually saturated) with larger values.

The memory capacity of reservoirs with multiple inputs (often many dimensions) was investigated by Hermans and Schrauwen [243] who extended the MC measure to multidimensional data by applying Principal Component Analysis (PCA) to the input data and calculating the MC measure for each principal component and then averaging over the total number of components. The principal components of the input were used in order to make the inputs uncorrelated so as to avoid any false measures of MC which may occur from correlated input patterns.

As the number of inputs was increased, it was observed that the reservoir's memory of each input pattern decreased as a result of the increased mixing of input signals that influence the reservoir neurons' activations. When a reservoir was presented with input signals of different variances it was observed that as the spectral radius increased more memory was given to the input channels with higher variance. This suggests that reservoirs may improve their performance if only the first few principal components of the input signal are presented as input to the reservoir, rather than the whole signal where noise and irrelevant information would also be learned by the reservoir. A rescaling of all the principal components of the input signal using the input variance may also be beneficial as it would remove the effect of the higher variance signals consuming more of the memory capacity of a reservoir.

A recent novel reservoir architecture, *Simple Cycle Reservoir* (SCR) [218], was shown to have a memory capacity very close to the theoretical limit as outlined by Jaeger [229] and discussed earlier in section 3.4.6. The SCR architecture differs from a standard reservoir in its creation and its reservoir topology. Rather than having a randomly connected reservoir, the reservoir of an SCR usually contains neurons connected in a cycle. Secondly, the values assigned to the input-to-reservoir weights ($\mathbf{W}_{res}^{inp}$) and reservoir weights ($\mathbf{W}_{res}^{res}$) were deterministically assigned using two approaches, either a decimal expansion of irrational numbers (in the study by Rodan and Tiňo [218] π was used), or the first N binary symbols of the logistic map $f(x) = 4x(1-x)$. This approach was shown to offer similar performance to a standard ESN, with the added advantage of being completely replicable as a result of its deterministic weights.

3.7 Non-linear mapping and memory capacity

As this chapter has shown, RC offers a simple solution to overcome the many problems associated with typical RANNs which have hindered their use in the analysis of real-world datasets, in particular time-series datasets for which they are ideally suited. The attraction of applying RANNs to time-series data comes from their ability to learn the relationship between previous and the current input and output data which can often be highly non-linear, making linear systems ill-suited. In RC, these two characteristics also apply to a reservoir, with the added advantage of simple and fast training.

Section 3.4 presented several parameters along with their effects on the dynamics of the reservoir. Of all the parameters of a reservoir network that can be varied, the two parameters which have the largest influence on reservoir dynamics, it could be argued, are the input scale and the spectral radius. As was shown in section 3.4.1 the input scale controls the impact of the input on the reservoir activations: the higher the scale, the higher the influence of the input, whereas the spectral radius controls the impact of the previous reservoir states on the current reservoir state where high values increase this impact. Higher input scaling and spectral radius (values over unity) decrease the memory capacity of a reservoir as a result of moving their working point to more non-linear regions of each neurons' activations where eventually their activations will become saturated. Paradoxically, for some highly non-linear time-series datasets, such as ones collected from real-world reinforced concrete structures discussed later in Chapter 5, a high input scale and/or spectral radius can offer the best performance. More typically, one would like to obtain an ESN whose reservoir has as high a memory capacity as possible, while also retaining the ability to perform non-linear mappings of

input signals in order to separate highly non-linear data well. Currently, however, it is not possible to tune these two parameters independently to simultaneously acquire an ESN reservoir with high amounts of these two characteristics.

This problem was briefly mentioned in one of Jaeger's first publications on ESNs [182] where in order to achieve a long short-term memory in a reservoir, it was recommended to use small input weights. Jaeger then conjectured that this may conflict with non-linear task characteristics where a larger input scale may be required. This problem was later hypothesised by Büsing et al [201] where the kernel quality and memory capacity of a reservoir with binary (spiking) and analog (whose activations could be based on various functions, such as *tanh*) were investigated. The kernel quality (introduced by Legenstein and Maass [237]) is measure of the reservoir's ability to represent different inputs, where low values mean that the reservoir maps all of the inputs to the same area in its state space and high values mean that the reservoir maps all of the inputs over a large area of its state space. Essentially, higher values are typically desired as the reservoir's ability to separate non-linear datapoints increases when its state space ranges over a larger area. In the study by Büsing et al it was hypothesised that linear networks such as networks with delay lines have a high MC but perform poorly on non-linear datasets, thus having a low kernel quality. The reservoirs studied in their work were able to offer good performance on the datasets studied, but had memory capacity capabilities of the order of $\mathcal{O}log(N)$ compared to $\mathcal{O}(N)$ as offered by linear networks [201].

This antagonistic trade-off between the non-linear mapping capabilities and the memory capacity of a reservoir was later investigated by Verstraeten et al [226] where the memory requirements of a task were found to be dominant over the non-linear map-

ping capabilities of a reservoir when addressing problems with high amounts of both (in this study, the extended XOR dataset introduced later in section 5.1.3 of Chapter 5 was used). This is due to the fact that a reservoir, which has a higher MC score as a result of its neuron's working points being in their linear regions, offers better performance than a reservoir which contains more non-linear neurons and as a result loses some of this memory [226]. This was shown using the deviation from linearity measure introduced in section 3.6.3 where, for tasks which required both large amounts of memory and non-linear mapping, reservoirs which were often tuned towards increasing their MC, rather than their non-linear mapping capabilities, were found to be preferentially obtained.

Figure 3.12 below shows the dynamical behaviour of a reservoir with different values of input scale and spectral radius, indicating the values which correspond to a value of $\delta_\phi = 0.8$ (dashed line) and $\widetilde{LLE}_{max}$ (dotted line). This shows that with high values for the input scale and spectral radius one obtains an unstable reservoir. With relatively low values of both parameters a reservoir which operates in the linear region is obtained, where the reservoir has maximal MC (as shown by Jaeger [182]), while slightly higher values of the input scale and spectral radius move the reservoir towards its non-linear regime which is comparatively smaller than the other two regimes.

Figure 3.13 below characterises the different regions of the *tanh* activation function that a neuron's working point can operate.

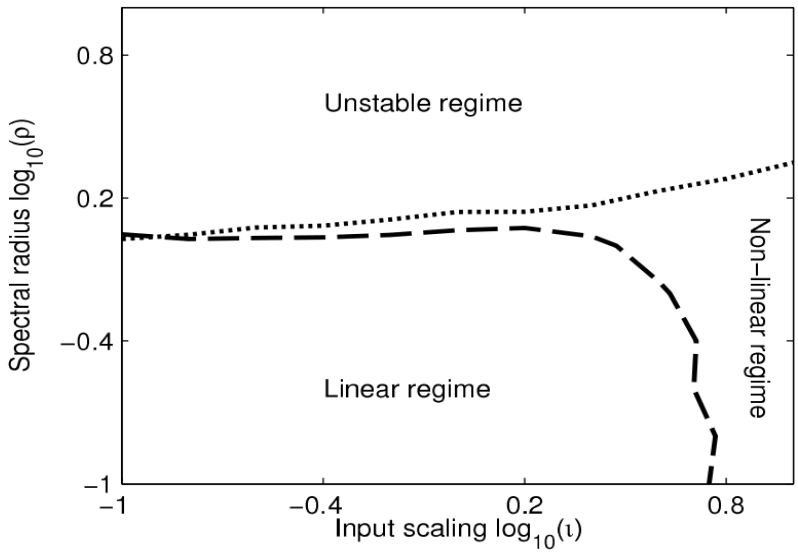


Figure 3.12: The different dynamical behaviours of a reservoir. The dashed line shows the different values of the input scale and spectral radius where $\delta_\phi = 0.8$, while the dotted line shows where $\widetilde{LLE}_{max}$ (reproduced from Verstraeten et al [226]).

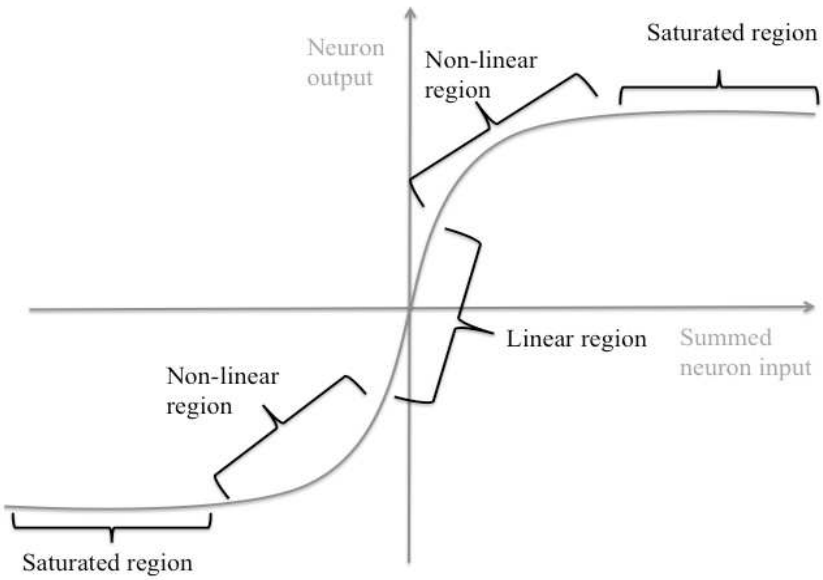


Figure 3.13: The *tanh* activation function and the different regions where its working point can be found.

Inspection of Figure 3.13 shows that the *tanh* activation function has effectively three regions in which a neuron's working point can belong. A neuron's working point will generally be in the linear region of the activation function given a small input signal and an input scale and spectral radius smaller than unity. Populations of neurons whose activations are in this region will create a reservoir with the highest MC, which can be maximised for small input scales and a spectral radius close to unity, as was shown in Figure 3.11 above. Conversely, the neuron's non-linearities as measured by δ_ϕ are towards their lowest values. As this is the linear region of the neuron's activation function, this is also the region where the neuron has the highest discriminatory power to distinguish between input patterns as a small change in the neuron's summed input leads to a large change in its output. As this region is approximately linear, a neuron

with a working point in this area will not handle non-linear data well, as it is less able to perform non-linear separation in its linear region.

In order to be better able to separate non-linear data, the neuron's working point must move towards its non-linear regions. As a result of moving out of its linear region, the MC of the neuron decreases. For very highly non-linear datasets, the working point of the neuron is often pushed into one of its saturation regions where it either stays for the length of the dataset or flips from one saturated region to the next (as was shown previously in Figure 3.4). In these regions the MC of the reservoir is the lowest, while the non-linearities present inside it as measured by δ_ϕ are the highest. A reservoir whose neuron's working points are saturated can also be indicated by an $\widetilde{LLE}_{max}$ value which is larger than zero, indicating potential instability problems.

When saturated, a neuron's ability to distinguish a difference between small changes in its inputs decreases since small changes in its summed inputs leads to a small change in its output. In the case of highly non-linear data which changes dramatically from time step to the next, this is not so problematic as the change in input is such to create a substantial change in the neuron's output. In this case the neuron maps one input to an area in its state space, while a different input will be mapped to a completely different area of its state space, enabling the non-linear separation of the highly non-linear input data over an increased state space onto which the input can be mapped. A typical indication of a highly non-linear dataset can be an input scale and/or spectral radius much higher than unity which enables best performance for some tasks, where little MC is perhaps required. This may offer an explanation as to why for datasets which are highly non-linear and require little MC, large input scales and/or spectral radius values which drive the reservoir neurons to their saturation regions may offer

best performance.

For datasets which are highly non-linear and require little MC (or which are very linear and require a long MC), this trade-off is not present within a reservoir as its neurons can move to their non-linear or saturated regions (or linear regions) and perform the non-linear mapping (memory of previous inputs) required from the task. For datasets which require large amounts of both, a trade-off between MC and non-linear mapping has to be made where MC is often given priority. In turn this reduces a reservoir's ability to perform highly non-linear mappings of the input data which can result in inferior performance should a reservoir which is able to perform these two requirements well be possible.

To further illustrate this trade-off, the deviation from linearity and the MC of a reservoir when increasing its input scale are plotted against each other in Figure 3.14. The input scale was incremented over the range 0.1 to 100 (using increments of 0.1 until unity was reached, after which the input scale was increased by one until a value of 10 was reached, after which an input scale of 20, 30, 50 and 100 was used).

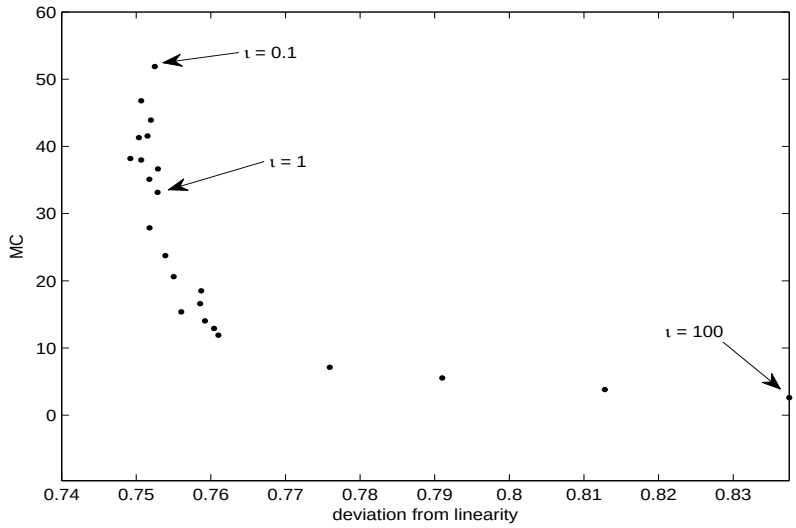


Figure 3.14: The effect of varying the input scale on the memory capacity and linearity of an ESN's reservoir showing a trade-off between the two characteristics (note that ρ is equal to unity here). Broadly, following the trend line from left to right each point shows the network characteristics obtained for increasing values of input scale where the non-linearities contained within the reservoir increase as shown by its deviation from linearity values (X axis) which, after values greater than unity, reduces the MC of the reservoir (Y axis) as the majority of reservoir neurons' working points are in their saturated regions.

As Figure 3.14 shows, there is a trade-off between the memory capacity and the non-linearity of the reservoir; the higher the MC, the more linear the reservoir; the more non-linear the reservoir, the lower the MC. The figure also shows that the highest MC is obtained with input scale values smaller than unity (as was shown by Jaeger [182]). Once the input scale of the reservoir exceeds unity its MC decreases dramatically while the non-linearities present inside its reservoir increase. In most real-world temporal datasets both memory and non-linear separation capabilities are required. Figure 3.15 below shows the effect of increasing the spectral radius (using the same values as for the

input scale in Figure 3.14) on a reservoir's short term memory and the non-linearities it contains as shown by the MC and deviation from linearity measures.

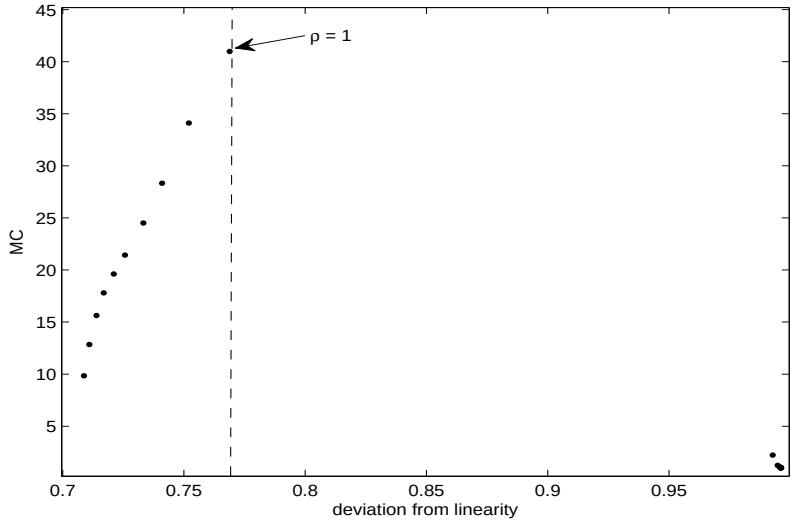


Figure 3.15: The effect of varying the spectral radius on the memory capacity and linearity of an ESN reservoir showing a trade-off between the two characteristics (note that ι is equal to unity here). Broadly, following the trend line from left to right each point shows the network characteristics obtained for increasing values of spectral radius where the non-linearities contained within the reservoir increase as shown by its deviation from linearity values (X axis). The MC of the reservoir increases as the spectral radius values approach unity after which values larger than unity have a detrimental effect on the reservoir's MC (Y axis).

Figure 3.15 shows that a value of unity for a reservoir's spectral radius gives the highest memory capacity when using an input scale of unity. The lowest value for the reservoir's deviation from linearity can be observed when its spectral radius is smallest at 0.1. This makes the reservoir activations very contractive, which means that their activations soon die out, reducing the reservoir's memory capacity and also any non-linearities it may contain. The figure also shows that to obtain the highest memory

capacity, the amount of non-linearities the reservoir contains must also slightly increase. After the non-linearities inside the reservoir exceed a certain point (around 0.77 in the figure), the reservoir becomes less able to recall inputs from the past, as shown by the vast reduction in its MC.

As a final investigation into the effect of changing ESN parameters on the MC and deviation from linearity of a reservoir, the leak rate was varied over the range 10^0 to $10^{-3.6}$ using a power increment of 0.4. This is shown in Figure 3.16 below.

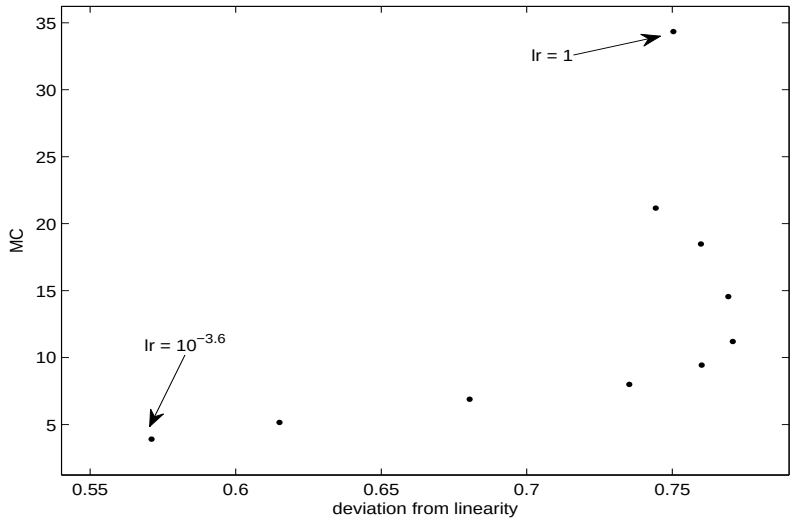


Figure 3.16: The effect of varying the leak rate on the memory capacity and linearity of an ESN reservoir showing a trade-off between the two characteristics (note that here both ι and ρ were equal to unity). Broadly, following the trend line from left to right each point shows the network characteristics obtained for increasing values of leak rate where the non-linearities contained within the reservoir increase before decreasing again as the leak rate approaches unity as was also shown previously in Figure 3.10. The MC of the reservoir increases as the leak rate values approach unity (Y axis).

Figure 3.16 shows that as the leak rate becomes smaller the MC of the reservoir decreases. This is somewhat counter-intuitive as smaller values of the leak rate increase the MC of the reservoir at a neuron level. However, the leak rate also acts a low pass filter which means that the increased MC obtained at neuron level is only valid for lower frequencies since the higher frequencies of the input are more highly attenuated. As the leak rate becomes smaller, too much of the input is filtered out and as a result the MC of the reservoir suffers, as shown towards the bottom left of Figure 3.16. As a result of filtering the input signal, the deviation from linearity of the reservoir decreases with smaller values of leak rate as the input signal contains fewer frequencies. As the leak rate increases, a maximal deviation from linearity is obtained when the leak rate is equal to 0.0251 (furthest point right), after which it decreases again as the leak rate approaches unity. The filtering effect of the leak rate can be countered by upsampling the input as was shown by Dutoit [227].

3.8 Applications

3.8.1 Time-series prediction

Although RC is a relatively new field, its use in different domains has been rather widespread, giving promising results. One of the first uses of ESNs was time-series prediction, as described by Jaeger [64, 182, 183] and Jaeger and Haas [184]. Such time-series include sinewave generation, the Mackey-Glass series and 10^{th} and 30^{th} order NARMA equations. Applied to the Mackey-Glass dataset, reservoirs were trained with different values of the delay parameter of 17 and 30, and compared to other techniques such as an MLP with output feedback [244] and a self-organising map combined with a local modelling technique [245]. It was found that reservoirs performed two orders

of magnitude better in terms of NRMSE than any other reported technique when the reservoir was required to give an output at a time step in the future with a delay equal to 17 and better than any other reported technique when using a delay of 30, predicting 84 time steps ahead.

Reservoirs and groups of smaller reservoirs were recently applied to monthly time-series prediction of electricity production, plastic and rubber goods production, metal goods production and machinery production by Wyffels and Schrauwen [211]. The authors showed that, amongst others, reservoirs offer a competitive solution for datasets which contain seasonal trends using seasonal decomposition to identify the trend cycle as well as seasonal and irregular components of the data. The three components of the data were modelled separately by the network and then recombined to give the prediction of the dataset. While the approach of using groups of reservoirs (referred to as a voting system) was more computationally expensive, for most datasets, this approach outperformed a Least Squares SVM (LS-SVM) [149] and the Census X-12-ARIMA program [246].

3.8.2 Speech and phoneme recognition

Reservoirs have been widely used in the speech recognition domain. Verstraeten [210, 247] used an LSM to classify spoken digits from zero to nine from a subset of the TI46 speech corpus [248] which contained spoken utterances of five female speakers (details of this task can be found below in Chapter 5). It was found that the LSM approach offered similar performance when compared to the state-of-the-art Hidden Markov Model (HMM) whilst being robust to noisy data and being well suited to continuous real-world data.

Schrauwen and Büsing [249] recently applied a modified reservoir approach, the Temporal Reservoir Machine (TRM), to the same TI46 dataset using a Restricted Boltzman Machine (RBM) combined with a reservoir. Essentially the reservoir was used to perform the temporal integration of the input data at each time step which was then used as input to an RBM. By combining these two approaches, the disadvantages of using conditional RBMs (CRBMs) and Recurrent Temporal RBMs (RTRBMs) were overcome.

CRBMs are able to process temporal data through the use of a model order parameter, m, which essentially controls the amount of memory the network has. This approach becomes problematic when applying CRBMs to data where inputs have dependencies on long time scales as large numbers of weights need to be calculated. RTRBMs contain recurrent weights which are trained using backpropagation through time and as a result, suffer from slow and sub-optimal convergence caused by local minima and bifurcations, which occur when a change of parameters leads to changes in a system's behaviour, from periodic to chaotic for example [250].

When trained on 300 random spoken digits and tested on 200 noisy spoken digits corrupted with an SNR of 5dB the TRM was found to offer an improved Word Error Rate (WER) (the proportion of incorrect utterances) score of 0.091, whereas a CRBM achieved a WER of 0.653. Dimensionality reduction using PCA was also investigated in this study where it was found to improve performance of the TRM giving a WER of 0.07. This improvement in performance was achieved as a result of reducing the number of inputs to the reservoir, which in turn allowed the reservoir to have a longer short-term memory capacity. This is due to the fact that large input dimensions often contain high rates of redundancy which drive the reservoir dynamics in random

directions, negatively affecting the reservoir's memory of previous inputs. This was also reported in a later study by Hermans and Schrauwen [243] which was briefly discussed above in section 3.6.4.

Hermans and Schrauwen [251] also investigated the performance of a modified reservoir approach using the TI46 dataset in which the input to reservoir and reservoir to output weights were trained using a simplified version of backpropagation through time (BPTT), rather than the traditional linear regression approaches, where the error was propagated back one time step, named one step backpropagation (OSBP). Compared to networks which were created randomly and trained with RLS on the digit recognition task the OSBP approach offered superior performance when tested on unseen clean digits with a small amount of neurons.

Separate studies by Skowronski and Harris [252, 253] applied a predictive and discriminative ESN to the whole of the TI46 speech corpus (all eight male and female speakers) dataset which were shown to be more robust to noise than HMMs. The predictive ESN [252] used a competitive winner-take-all network of readout filters between the reservoir and output layer. These filters were grouped together to form a state which can be moved from left to right during training and testing. The readout filters were trained using a variation of the segmental K-means algorithm [254] where the optimal sequence of filters was found by minimising the mean squared error between the target and actual output of the reservoir. Both the predictive ESN and HMM classified almost all the test patterns correctly with a 1% error rate, while the predictive ESN outperformed the HMM by 8 ± 1 dB SNR when applied to corrupted test data with additive speech-shaped noise (generated by calculating the average power spectral density of all of the utterances with a 100-tap FIR filter which was then used to filter

Gaussian white noise using the same sampling rate as the spoken utterances).

In their later study, Skowronski and Harris [253] altered the approach of the predictive ESN to produce a discriminative ESN such that rather than setting the target value to that of a future input, the output layer was constructed using a one-of-many output encoding where each word was represented by an output channel. When applied to the same test set as in their earlier study, the discriminative ESN was more robust to noise by 7 dB than the predictive ESN and 15 dB when compared to an HMM.

RC was recently applied to phoneme recognition [255] where it was shown to offer similar performance to state-of-the-art HMMs. Here a hierarchical approach was adopted where two reservoir layers were used, both of which contained 20,000 neurons. This was adopted to overcome the problem found in single layer reservoirs where the number of competing outputs is high and impacts negatively on the performance of the reservoir. By using a second layer, the margin between the output of the winning neuron and its competition was much higher, which, in the case of phoneme recognition increased performance.

In a related field, reservoirs were applied to the domain of non-linear audio processing [256] to solve three tasks. The first involved the system identification of a tube amplifier plugin using two different plugins as datasets. The second task involved non-linear audio prediction of a signal in order to achieve audio restoration when noise is present or part of the signal is missing. The performance of the ESN with filter neurons and a *delay&sum* readout for this application was compared against a pattern matching algorithm and a linear autoregressive model. The third task involved the extraction of a melody from a polyphonic piece of music where a reservoir was compared to an

SVM. In all three tasks it was found that the reservoirs performed better than the state-of-the-art techniques for each application.

3.8.3 Robotics

Reservoirs have proved rather popular and successful in the robotics field. For example, Antonelo et al [257] used an ESN to solve the T-maze task. This task required a robot to navigate around a T-shaped maze to reach the correct goal on either the left or right arm of the T depending on a previously received signal or sign. Two environments were created, A and B, where B had a corridor which was twice as long as A's.

Two configurations of robot were used: a robot fitted with three colour and three distance sensors, and a robot fitted with seven colour and seven distance sensors. The outputs of the reservoir were two neurons which related to the robot's turning angle and speed. After initial training, both robot architectures were able to navigate the maze based on current sensor data and previous information. The ESN architecture with seven sensors for both colour and distance was more successful than the smaller architecture by 37% and 45% for environments A and B respectively. Added noise of 5% was found to reduce performance, especially for environment B which was harder to navigate due to the larger delay in turning sign and required response from the reservoir. Other work using reservoirs and robotics can be found elsewhere in the literature for example [258, 259, 260, 261, 262, 263, 264, 265, 266, 267].

3.8.4 Environmental anomaly detection

ESNs have been applied to anomaly detection using real-world sensor networks. In one study [268], an ESN was used to detect anomalies in underground coal mine sensor data in order to give warning of critical gas concentrations to prevent dangerous explosions and reduce the number of costly false alarms. The ESN used was compared to a benchmark Bayesian network and was shown to give an overall performance similar to the Bayesian network. When given multiple sources from multiple sensors, the ESN gave the same performance as with the single-source data, making them well suited to anomaly detection as the data analysed often contains variability arising from multi-sensor readings of multiple gases.

Chang et al [269] also used ESNs for anomaly detection in temperature and humidity data collected using a sensor network. In this work, the ESN was implemented on the sensor and trained using normal signals in which the reservoir was tasked with predicting the next datapoint. Once trained, the reservoir was used in real-time and required to predict the next data point based on its previous inputs; a significant difference between the prediction and the actual data point was used as an indication of the presence of an anomaly. It was found that the ESN approach not only offered superior performance compared to standard threshold-based anomaly detection algorithms, but it was also able to differentiate between different types of anomalies. The fact that different anomalies could be detected in this study meant that RC could potentially be a useful approach for defect detection and identification of each defect type when applied to EMAD data collected from reinforced concrete structures as outlined in Chapter 1 previously. In another study [270], ESNs located on every sensor in a network were used to detect faults within the sensors themselves where it was found to perform well,

even in events where half of the nodes failed.

3.8.5 Other applications

Devert et al [271] presented an unsupervised ESN which was used to solve a classical benchmark problem from multi-cellular artificial embryogeny containing genotypes and phenotypes where an evolutionary approach was used to optimise the reservoir-output weights of an ESN. Essentially this task involved cells placed on a two-dimensional grid. Each cell had a controller (genotype, which was the ESN in this case) which determined its state and the amount of chemicals emitted at any time step (phenotype, represented by a grey-scale level). The goal of the cells was to reach a target image in a given number of time steps. This approach was chosen as it scaled well when the search space became high dimensional with large reservoir sizes. The performance of the ESN to replicate two target images was compared to the state-of-the-art neuro-evolution method, NEAT [272]. The ESN was found to outperform the NEAT algorithm on the replication of the first image, while no significant difference was found between the two techniques on the second image. In addition the reservoir was found to converge faster and have a smaller variance than the NEAT approach.

Noris et al [231] presented work using an ESN to classify the walking gait of autistic and healthy children where best classification accuracy of 80% was obtained. The authors also investigated the effect of varying three network parameters: network connectivity, spectral radius and input scaling. Whilst the connectivity had little effect on the ESN's performance (as was discussed earlier in section 3.2), input scaling was found to make a difference of roughly 20%, while changing the spectral radius led to a slight improvement in performance where an optimal value greater than unity was found.

A reservoir where the input weights, as well as the output weights, were trained was investigated by Frank and Jacobsson [273] in the domain of sentence processing to predict the next word in a given sentence where previous studies [274, 275, 276] have reported mixed success. The input weights were updated to implicitly encode words of the same category (for example, nouns) such that their representations clustered together. The reservoir was then trained to give a probability estimate of the next word in a sentence based on a word's category. The reservoir with trained input weights was found to offer better performance than a standard reservoir on both training and test sets.

An ESN was combined by Rao et al [277] with a sparse linear mapping of the reservoir activations for the prediction of primate hand movement based on neuron firing patterns. This approach overcame the problems typically associated with linear models, which struggle with the large disparity of hand movement data which results in poor performance, and the problems associated with other RANN approaches which, as discussed previously in Chapter 2 and above in this chapter, suffer from increased training complexity. The combination of an ESN with a sparse matrix also overcomes the problem of over-fitting when using large reservoirs with low dimensional input patterns. The performance of the ESN combined with a sparse linear mapper was found to match other existing techniques, but with the advantage over the other techniques of quicker training times due to a less complex model and fewer weight updates.

Other domains in which reservoirs have been applied include, amongst others, monitoring of multimachine power systems with an ESN Wide Area Monitor system [278], motor control of human electrocorticogram neuroprosthetics [279], bit error rate prediction for mobile wireless networks [280], the analysis of the toxicity of chemical com-

pounds [281], short-term load forecasting for power systems [282] and the real-time detection of epileptic seizures in Electroencephalograms of rats [283, 284].

3.8.6 Summary of applications

In summary RC, despite its relative infancy and remaining open research questions, has been applied to a variety of problem domains with promising results. In many cases, RC approaches have offered, at worst, comparable performance to the state-of-the-art techniques from the relevant domains. The simple and fast training algorithms of RC techniques has often been quoted as their main advantage over other techniques. As a result of their simple training, RC techniques have also been shown to outperform gradient descent-based training algorithms. As a reservoir has recurrent connections, RC is well-suited to time-series data analysis, which is clear from the number of time-series domains to which it has been applied. A major motivation for the investigation into the use of RC for the automatic detection of defects in EMAD data collected from reinforced concrete structures is that it has already been successfully applied in anomaly detection using sensor networks [268, 269, 270]. This suggests that with its ability to map the relationship between inputs and outputs over a period of time and with its simple training approach, RC can be successfully applied to detect anomalies, or defects, within EMAD data. The results of which are given below in Chapter 6.

3.9 Summary

This chapter has surveyed the field of reservoir computing (RC) as a recent addition to the field of recurrent neural networks. While established recurrent neural network architectures possess the capability of learning to map inputs and outputs contained

over various periods in time-series data as a result of their short term memory and non-linear mapping capabilities, their use in real-world applications has been limited by their slow and complicated training regimes. As reservoir training involves only altering the reservoir-to-output weights using a simple linear regression approach, RC offers a solution to this problem. While reservoirs are easy to train, there are some parameters that require tuning which are typically found by performing a large parameter sweep over a range of possible values, although reservoir adaptation has recently been introduced to alleviate this problem somewhat. Tuning these parameters can drastically alter the dynamics of the reservoir which was the focus of the latter part of this chapter, along with the affects changing these parameters has on the dynamics of the reservoir.

Perhaps the two most important capabilities of a reservoir are its ability to remember the relationship between previous inputs and target outputs and to project the input data onto a high dimensional state space from which hyperplanes can be drawn to separate datapoints from different classes (the same also applies to regression problems, but classification problems are used to illustrate this point). While a reservoir's datapoint separation capabilities can increase with a larger memory capacity (for example, a task which requires memory of the last 10 inputs will have high separation capabilities if it can remember all 10 inputs and classify them correctly), for datasets which require high amounts of memory and are very non-linear it has been shown that the two capabilities behave as an antagonistic trade-off. This antagonism arises as a result of the reservoir neurons' activation function. In a reservoir whose neurons' activations are around their origin they have maximum memory capacity, but their working point is in the linear part of the activation function. When processing highly non-linear data, which changes dramatically from one time step to the next, the reservoir must be

able to map each input to a separate region of its state space in order to classify each data point correctly. In order to do this successfully, the reservoir's neurons should operate in their non-linear, or sometimes even their saturated regions, in order to map the input data over a larger state space. In extreme cases (for example, for reservoirs whose neurons mainly operate in their saturation zones) this can also lead to unstable reservoir characteristics. This ability is compromised when dealing with data which contains high amounts of non-linearity and short term memory as the reservoir is often tuned towards maximal memory capacity as this gives best performance. Conversely, when the majority of the reservoir neurons operate in the non-linear region of their activation, their ability to remember inputs decreases. The next chapter introduces a novel extension to the ESN architecture that offers a possible solution to this problem which is *Reservoir with Random Static Projections*.

Chapter 4

A method for overcoming the trade-off between non-linear mapping and memory capacity

4.1 Introduction

The previous chapter introduced Reservoir Computing (RC) as a relatively recent and promising field of Recurrent Artificial Neural Networks (RANNs) where a dynamic, randomly connected reservoir of neurons was used to project the input data onto a rich and multidimensional state space. A linear regression technique is then applied to the reservoir, drawing hyperplanes between datapoints of different classes and changing the reservoir-to-output weights. This is the only weight update to take place during training. This simple yet effective approach to RANN training has enabled reservoirs to equal, and in some cases outperform, many state-of-the-art techniques as was also surveyed in the previous chapter.

Although RC has been successfully applied to a variety of domains, as a result of its infancy many research questions still remain unanswered. While many of these concern issues such as the types and parameters (if any) of reservoirs which are best suited for certain types of tasks, an important aim of this research was to investigate the memory and non-linear mapping capabilities of a reservoir when applied to datasets which require the reservoir to possess both characteristics in substantial quantities. Currently however, a single reservoir cannot possess high levels of both of these characteristics, as shown in the previous chapter. There is an antagonistic trade-off: a reservoir can have either a high memory capacity and as a result low non-linear mapping capabilities, or high non-linear mapping capabilities and a resulting low memory capacity. While this is not problematic for datasets which require high degrees of one characteristic and less of the other, it can become problematic for real-world time-series datasets which might require high quantities of both. Indeed, this situation was observed when applying a reservoir to data collected from a real-world reinforced concrete structure (see section 5.2.2 of Chapter 5, for further details on this dataset). Here values of the spectral radius and, in particular, the input scale which enabled best performance were high. Preferred input scales found from extensive grid searches ranged from 20 to 50, resulting in highly saturated reservoirs which had a negative impact on their classification capabilities (for more details on its performance on this real-world dataset, see Chapter 6 below). This chapter presents a novel extension to RC architectures that overcomes this trade-off: Reservoir with Random Static Projections (R^2SP) [285, 286]. But first this chapter describes the *extreme learning machine* [287] whose input data can be presented via a moving window to give a time-delayed ELM (TD-ELM).

4.2 Extreme Learning Machines and their extensions to date

The ELM [287] approach consists of a standard feed-forward network with an input, a hidden and an output layer of neurons in which the input weights and bias to the hidden layer are randomly selected. Using an ELM approach it was shown, provided that the activation functions of the hidden neurons were infinitely differentiable (for example, sigmoidal, radial basis, sine, cosine and exponential activation functions), that the ELM approach outperformed other approaches including an MLP and SVM on some benchmark tasks [287]. Since only the hidden layer to output layer weighted connections are trained using linear regression (in most studies the Moore-Penrose pseudoinverse is used), the ELM can be viewed as a kind of memory-less reservoir.

ELMs have since been extended by Miche et al [288] to give an Optimally Pruned-ELM (OP-ELM) which uses a standard ELM with additional hidden layer neurons and prunes away the least significant hidden layer neurons based on their contribution to the network's output error. The ELM's error is calculated using the Least Angle Regression (LARS) [289] or multiresponse sparse regression (MRSR) [290] and Leave One Out (LOO) techniques which are outlined later in this chapter. A further alternate approach was later presented by Feng et al [291] as the Error Minimised ELM (EM-ELM) where neurons (or groups of neurons) are added to the ELM (rather than pruned) with an output weight recalculation after each addition. This variant has since been extended as an Enhancement of EM-ELM (EEM-ELM) [292] in which several candidate neurons were created at each addition and the neuron which led to the greatest reduction in error was retained and added to the network: resulting in a more compact network [292]. A similar variant of the EM-ELM, Convex Incremental ELM (CI-ELM) [293], has also

been introduced where the output weights were recalculated after each addition of a new neuron. It was also shown in this study that the CI-ELM was a universal approximator when using activation functions such as radial basis and sigmoid functions.

The evolutionary ELM (E-ELM) was adopted by Zhu et al [294], where an evolutionary approach [295] was used to find the optimal input weights and the biases of the hidden layer neurons. The E-ELM was found to give much more compact networks as well as offer performance that outperformed or equalled a standard ELM when applied to four classification tasks. Recently, ELMs have also been applied to SVMs where it was shown that SVM kernels can be replaced with ELM kernels to give improved performance [296, 297]. While SVM and ELM approaches share similar characteristics, ELMs offer improved performance and require less optimisation constraints, resulting in faster training times due to fewer parameters to optimise [298]. An online sequential fuzzy ELM was investigated by Rong et al [299] where it was shown to offer similar, and in some cases significant performance in several benchmark problems such as: non-linear system identification [300], regression and classification with the added advantage of reduced training times.

ELMs have been applied to numerous problem domains including ordinal regression (where the ordinal relationship amongst classes is used) by Deng et al [301] who showed that as a result of fast training, their ability to handle non-linear data and learn either in batch mode or online, three custom ELM approaches offered superior performance to Gaussian Processes for Ordinal Regression and SVMs for Ordinal Regression (ORSVM). Other applications of ELMs include the classification of EEG signals [302], object recognition and classification in images [303], multi-class classification problems [304], the classification of gene expressions for the diagnosis of cancer [305] and the

classification of music genres [306].

By combining an ELM with a time delayed moving window of the input data, similar network behaviour should be obtained as hypothesised below for the R^{2}SP architecture. The moving window of the input data would provide the network with memory of its previous input, while the network's hidden layer of feedforward neurons would perform the non-linear transformation of the data. Using this approach, the window size (i.e. the memory length of the input data) would require tuning, although given the fast training of the ELM approach this would be possible to achieve with relative ease. For datasets which contain high amounts of non-linearity and memory, the R^{2}SP and time-delayed ELM architectures may therefore offer improved performance.

4.3 Extreme Learning Machines and time-delayed inputs

An Extreme Learning Machine (ELM) with a time-delayed input (TD-ELM) in order to give it short-term memory of its input was investigated. This was implemented using a sliding window of the input data of length N as input to the ELM. Each datapoint was presented to an input neuron of the network, giving it a total of $N \times K$ neurons where N is the length of the window and K is the number of input layer neurons that would have been used in a network without a time-delayed input. At each time-step, the window was shifted through the input data by one time-step. Presenting a finite window of input data is one of two techniques that can be used in machine learning to allow a technique to contain a temporal representation of input data. The other commonly used approach is a recurrent technique, such as a reservoir, which possesses a fading temporal representation of the input data. A schematic representation of this

implementation is shown in Figure 4.1. On the left is an ELM with three input neurons, three hidden layer neurons and one output neuron. Using a time window of two time steps the TD-ELM (right) has a total of six input neurons, which are presented with the three data points from the current time step, as well as the three inputs from the previous time step.

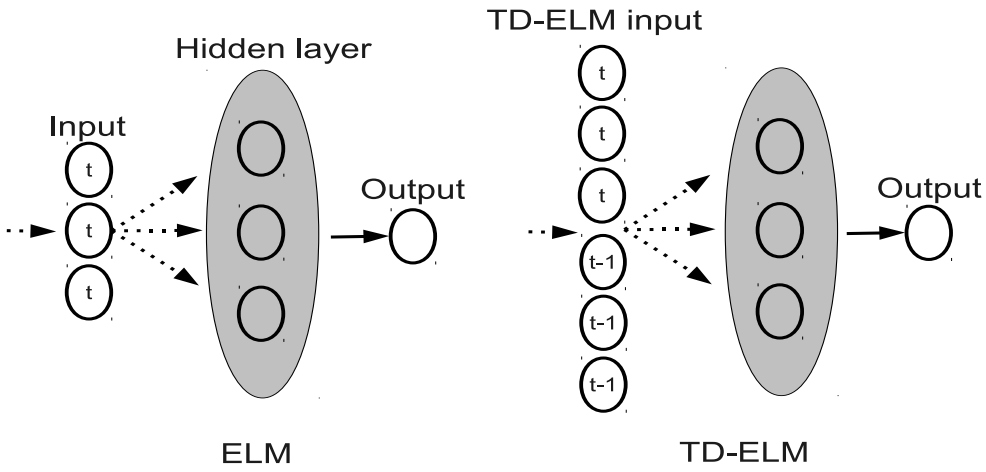


Figure 4.1: An ELM (left) and a TD-ELM (right) with a delay line as its input. The delay line transforms the input data into groups of values determined by a window size which are then shifted by one time-step. For example, with a window size of two, the first datapoints presented at time t will consist of data from time t and $t-1$, while the data presented at time $t+1$ will consist of data from time $t+1$ and time t. This repeats until the end of the dataset is reached. The solid arrows indicate connections which are modified during training, while the dashed arrows are weighted connections initialised at network creation and remain unaltered.

Recently some studies have investigated the use of a moving window in ELM networks for time-series data analysis. For example, van Heeswijk et al [307] used an ensemble of

ELMs for time-series prediction. The different ELMs were trained on the original input data (i.e. not using the moving window) which were then compared against ensembles of the same configuration using a sliding window and a growing window (i.e. all of the input data presented so far) of the input data. On a well-known stationary (i.e. the training data represents the test data well) dataset (the Santa Fe Laser time-series [308]), a growing window was found to offer the best performance, followed by the original dataset with no windowing. The use of the sliding window on this dataset gave the worst performance. When applied to a non-stationary (i.e. the test data can be different from the training data for which the system must retrain and adapt as new data points arrive) dataset (the Quebec Births time-series[1]) presenting input to the network using a sliding window gave the best performance, followed by the growing window approach and the original dataset.

An ELM using a moving window of its input data was applied to several datasets including the Mackey-Glass and Santa Fe datasets by Singh and Balasundaram [309] where the sine (an activation function characterised by a sine wave) and sigmoid activation functions gave the best results in comparison to the hard limiter function. Chen and Ou [310] also used a moving window of the input data to forecast sales using historic data as the input. The TD-ELM approach reported in their work was found to outperform several techniques including a backpropagation trained MLP, while also having the lowest training time.

As the TD-ELM has been investigated in the work outlined above, this research makes no claims towards the novelty of the TD-ELM approach. This is the first time presenting a moving window of input to an ELM has been termed a TD-ELM, but

[1]available from `http://robjhyndman.com/TSDL`

this approach has been used elsewhere in the literature in the past without a specific name. It is presented here as a comparison against the novel R^{2}SP architecture as an alternative approach to separating short-term memory and non-linear transformation.

4.4 Reservoir with Random Static Projections (R^{2}SP) architecture

As ELMs use the same simple training approach as RC and have been successfully applied to a variety of problem domains, it was conjectured that ELMs and RC could possibly complement each other. By combining the two approaches to form the R^{2}SP architecture, a reservoir could be tuned to perform the memory mapping of the data (due to its recurrent connections) while two ELM layers could perform an instantaneous non-linear transformation of the input data and the reservoir states, thereby overcoming the trade-off between the two abilities in a single RC network and enabling high amounts of MC and non-linear transformation to exist simultaneously within a single network.

As the R^{2}SP combines a reservoir with an ELM, the activations of the reservoir neurons are obtained as outlined in Equation 3.3 above (the reservoir of the R^{2}SP is denoted as R^{2}SP-reservoir). One layer of feedforward (static) neurons of the ELM were added in parallel with the R^{2}SP-reservoir to give an instantaneous non-linear mapping of the inputs from the current time step and is denoted as $stat_1$. Another layer of feedforward neurons was added between the R^{2}SP-reservoir and the output layer to give an instantaneous non-linear mapping of the R^{2}SP-reservoir at each time step which is denoted as $stat_2$. Figure 4.2 shows a schematic overview of the R^{2}SP architecture.

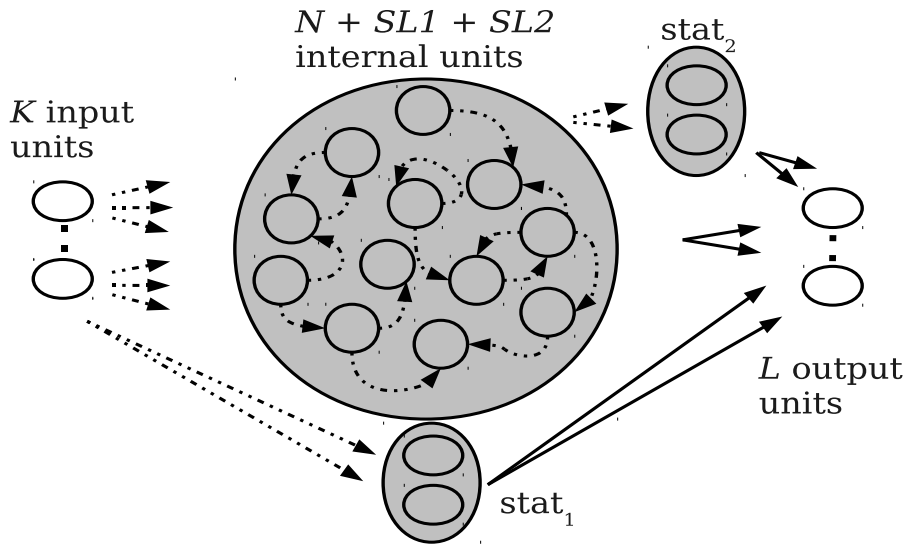


Figure 4.2: The R^{2}SP architecture. SL1 and SL2 denote the number of neurons in static layers $stat_1$ and $stat_2$ respectively. The solid arrows indicate the weighted connections which are modified during training. The remaining connections (as indicated by the dotted lines) are left unmodified after network initialisation.

Since $stat_1$ provides an instantaneous non-linear mapping of the input data it was expected to perform the majority of the non-linear mapping of the input data onto a higher dimensional state space. As the reaction of $stat_1$ neurons to a change in input data is instantaneous (compared to a reservoir which can take some time to react to changes in inputs as a result of its recurrent connections), $stat_1$ was also expected to help improve performance of the network when presented with fast changing input data.

The role of $stat_2$ was to map the R^{2}SP-reservoir activations onto a higher dimensional and more non-linear state space than the reservoir alone could achieve (if best performance of the R^{2}SP network were to result in the R^{2}SP-reservoir being tuned

towards maximum MC possible, then its state space would not be highly non-linear). The advantage of further transforming the reservoir activations onto a subsequent non-linear state space is that it aids the readout layer in separating datapoints which are mapped onto $stat_2$, a similar approach which was recently investigated by Gallicchio and Micheli [202]. In their study a static feedfoward layer was added between the reservoir layer and the readout layer to give a φ-ESN. All of the reservoir neurons were connected to the added static layer. The output weights were then calculated based on the activations of the static layer. By transforming the reservoir activations onto a higher dimensional representation it was shown that a φ-ESN outperformed a standard ESN. The φ-ESN approach differs from the newly proposed R^{2}SP approach as the R^{2}SP output weights are calculated based on the activations of the reservoir neurons, as well as $stat_2$'s neurons (which perform the same role of the static layer of the φ-ESN) and the neurons of $stat_1$.

The three sets of activations: the R^{2}SP-reservoir in addition to $stat_1$ and $stat_2$ are combined in order to calculate the output weights using ridge regression. The activations of $stat_1$ and $stat_2$ are defined as:

$$\mathbf{x}_{stat_1}(t) = f(\mathbf{W}^{inp}_{stat_1}\mathbf{u}(t)) \tag{4.1}$$

$$\mathbf{x}_{stat_2}(t) = f(\mathbf{W}^{res}_{stat_2}\mathbf{x}(t)) \tag{4.2}$$

where $\mathbf{x}_{stat_1}(t)$, $\mathbf{x}_{stat_2}(t)$, $\mathbf{W}^{inp}_{stat_1}$ and $\mathbf{W}^{res}_{stat_2}$ are the activations and input weight matrices of $stat_1$ and $stat_2$ respectively. All other symbols are as in Equations 3.3 and 3.4. The weights for the matrices $\mathbf{W}^{inp}_{stat_1}$ and $\mathbf{W}^{res}_{stat_2}$ are randomly chosen from a uniform distribution between -0.05 and 0.05.

A recent paper by Li et al [311] compared an ESN with an ELM when applied to six non-linear time-series prediction tasks. Overall it was found that the ESN offered the best performance when trained with small training datasets (typically 320 input patterns or less) and was less sensitive to changes in the number of neurons inside its reservoir. While the ELM's performance was more sensitive to the number of neurons in its hidden layer, it offered faster training times and better generalisation when applied to larger datasets which had more than 4,000 input patterns or more than the ESN. The results from the paper by Li et al suggest that the TD-ELM may outperform the ESN for the majority datasets (those with more than 320 input patterns) described in Chapter 5 of this thesis as a result of possessing higher amounts of memory of previous inputs and non-linear transformation than the ESN. Its performance against the R^2SP approach was estimated to be similar as both architectures can contain high amounts of non-linear separation and memory capacity in slightly different ways. A comparison between these three architectures when applied to these datasets is described later in Chapter 6.

4.4.1 Further tuning of reservoirs, extreme learning machines and R^2SP

The previous chapter introduced many network parameters that can be tuned in order to improve a reservoir's performance. Amongst these was the size of the reservoir. Another approach to tuning the size of the reservoir which provides best performance is to set the size at a value found through initial searches and then prune the least significant neurons from the reservoir (based on their contribution to the network's performance). While initial searches may provide a rough estimate as to the size of a network that provides reasonable performance, pruning may improve performance.

Furthermore, it can provide insights into the dynamics of the task and the network when applied to this task, which is also beneficial.

Pruning of ESNs has been reported by Dutoit et al [312, 313, 314] who showed that pruning of reservoir neurons helped to improve performance for some tasks as it regularises the reservoir (regularisation was introduced earlier in section 3.3.2 in Chapter 3). Pruning is also beneficial as it reduces the amount of computation required once the best size is found after the pruning procedure since there are less output connections to re-calculate, although finding this optimal size is computationally expensive. This is useful from a hardware implementation point of view where smaller networks will require less resources. It can also give further insights into the most useful output connections of the network, from which some conclusions can be drawn about the dynamics of the network and the requirements of the dataset. For example when using the R^2SP architecture, if, for a particular task, the majority of removals from the network are from the layer of $stat_1$, one could conclude that the network requires mainly memory from the R^2SP-reservoir in order to perform well and that non-linear separation of the input signal is not so important. In this case it could be that the dataset requires separation only along a few hyperplanes. Conversely where the majority of R^2SP-reservoir neurons are removed, one could argue that the network requires little memory from the R^2SP-reservoir and that non-linear separation provided by $stat_1$ is the most important characteristic of the network. From these observations, one can also gain insight into the characteristics of the dataset under investigation (whether data requires high non-linear separation or long-term memory capacity, or in some cases, where data requires high amounts of both).

4.4.2 Investigating the effects of pruning

The work conducted in this research has investigated the effect of pruning on the performance of the R^{2}SP by adopting the underlying principles of the OP-ELM approach. OP-ELM uses a combination of Least Angle Regression (LARS) [289] (or multiresponse sparse regression (MRSR) [290]) and Leave One Out (LOO) to rank and remove the least significant neurons. In this study, MRSR was used as the ranking algorithm which is an extension of LARS. Once the MRSR algorithm is complete (adopted from the OP-ELM Matlab toolbox [315]), a vector containing neuron numbers ranked from most significant to least significant is obtained. The error of the full network (i.e. without any pruning) was then calculated using cross validation with a grid search for the optimal regularisation parameter. During each fold, the output weights were trained using ridge regression as outlined in the previous chapter. The training and testing error of the network were then calculated after removal of the least significant neuron using the same approach as above. This process was then repeated until one neuron remained or the error increased significantly (for the investigative studies here, all but one neuron were removed from the network). The network configuration with the lowest test error was then selected as the optimal network and the number of neurons remaining in each layer calculated. To clarify, the pruning of neurons was performed as shown in the following algorithm:

1. load training and testing datasets;
2. calculate the R^{2}SP-reservoir, $stat_1$ and $stat_2$ activations when presented with training data;
3. combine activations of the R^{2}SP-reservoir, $stat_1$ and $stat_2$ into one matrix, $\mathbf{X}$;
4. perform MRSR (see below for a brief description of the MRSR algorithm) on

all activations which returns a matrix with the neurons ranked according to their significance to the error of the network where the last neuron is the least significant;

5. for neuron $i = 0{:}N-1$
    - remove neuron i from the network by setting its activations to zero where the first neuron is the least significant according to the MRSR algorithm and N is the total number of neurons;
    - calculate the output weights of updated network;
    - calculate the train and test error of the network;
6. end neuron iteration (return to step 5).

4.4.3 The MRSR algorithm

The MRSR algorithm is outlined in this section for clarity, although a full mathematical proof is not given. Readers are pointed to [290] for the full mathematical proof. The algorithm starts with a model of $\mathbf{Y} = \mathbf{XW}$ which aims to replicate a target matrix $\mathbf{T}$ as closely as possible. MRSR then ranks the reservoir neurons in order of significance to performance. The matrix $\mathbf{W}$ is initially a zero matrix and $\mathbf{X}$ is a matrix consisting of the activations of the R^{2}SP-reservoir, $stat_1$ and $stat_2$ neurons. The correlation between the residuals $(\mathbf{T} - \mathbf{Y})$ and the activations of each neuron is calculated, where the neuron with the highest correlation is added to the model. The corresponding row of the weight matrix $\mathbf{W}$ is then updated and the index of the neuron is recorded. This process repeats iteratively until all of the neurons have been added to the model $\mathbf{Y}$. For the work presented in this research, the ranked index of neurons is used to remove the

least significant neuron from the reservoir first. As outlined above, the output weights are then recalculated and the train and test error of the network is calculated.

The calculation of error relating to the removal of a neuron from the network differs slightly from the approach used in OP-ELM. In their OP-ELM work, the authors used PREdiction of Sum of Square (PRESS) to calculate the error, whereas this work calculates the error of the network after the removal of a neuron during cross validation for each network configuration. While this may take longer, particularly for larger datasets and network sizes, it gives a more reliable score as the PRESS results can be influenced by an accidental poor choice of training and test patterns. In such cases, for example, a testing set which is easy to classify would enable the network to give a high classification rate when actually it performs poorly on more difficult test datasets or subsets of test data which are more representative of the whole dataset.

4.4.4 Matlab implementation of R^{2}SP in the RCToolbox

The empirical work conducted in this research regarding RC and R^{2}SP has used the RC Toolbox version 1[2] (first introduced in Verstraeten et al [200]) which is programmed in Matlab. This well-used toolbox was created by the group based at the Reservoir Lab at the University of Gent, Belgium. An overview of this version of the RCToolbox can be found in the appendices of Verstraeten [210] and also in the RCToolbox manual [316]. For this research, aspects of the toolbox were updated to incorporate the added memory-less layers introduced to create R^{2}SP. While a thorough description of each part of the simulation is outside the scope of this document, a brief introduction into the processes involved in a simulation will be given in this section, as well as how

[2]Freely available from `http://www.elis.ugent.be/rct`

the R^{2}SP approach was implemented to fit into this toolbox. Figure 4.3 shows a flow diagram of a simulation in the RCToolbox.

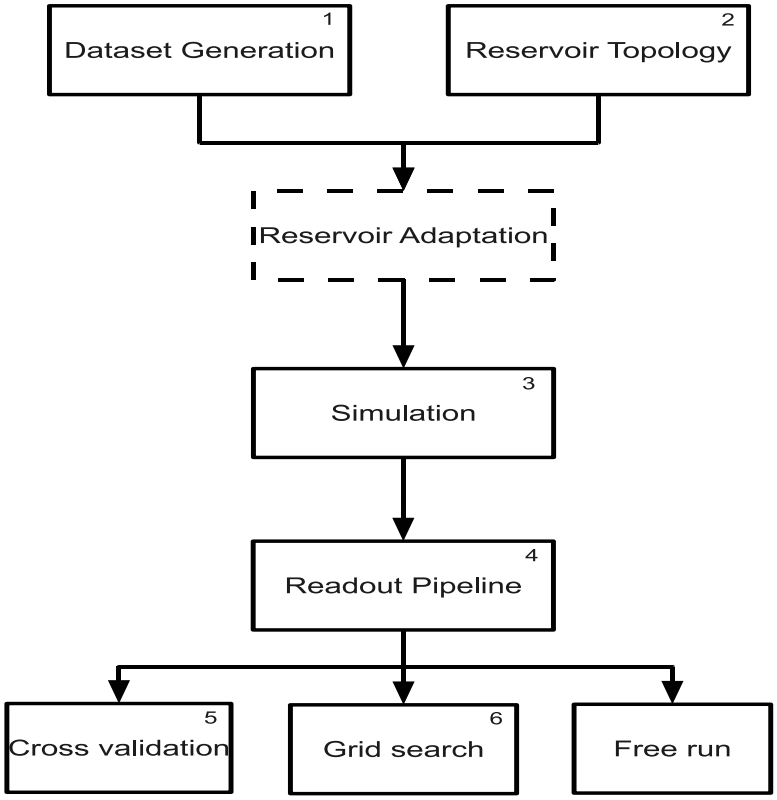


Figure 4.3: A flow diagram of a simulation in the RCToolbox (adapted from the RCToolbox version 1 manual [316]). The numbers in the top right corner of some boxes indicate modifications introduced into the code to accommodate the R^{2}SP approach, which are discussed in Table 4.1 below. The variables defined by the configuration file are loaded prior to boxes 1 and 2. Reservoir adaptation is optional as denoted by the dashed box.

As Figure 4.3 shows, a simulation (one reservoir simulated and trained for all data points of a dataset) consists of several steps. Firstly, one specifies the network param-

eters and the dataset on which one wishes to perform some analysis in a configuration file (parameters which are not defined by the user are given a default value). The reservoir is then created and any pre-adaptation (such as IP introduced in Chapter 3) is performed. In the simulations conducted in this work, no pre-adaptation was performed. The reservoir is then simulated. This involves the presentation of each input at each time step in order to calculate the activations of the reservoir neurons. In this work all training was performed using the batch update approach, where the reservoir activations were calculated for all input patterns in the time-series training dataset and the reservoir-to-output weights were calculated in a one-shot fashion. (Online training of the weights can also be computed, where the weights are calculated at each time step.) Once this is complete, the readout layer is trained using either cross validation, grid search or free run. In each of these approaches the error rate of the reservoir is calculated according to a score function specified by the user. This process is repeated for each parameter setting which can be defined in a configuration file. As a result of the random weight initialisation of each reservoir, each reservoir with specific parameters is usually simulated several times (in this research 100 times).

In this research, the readout function was mainly trained using the grid search approach, except in one case of the spoken digit recognition task which section 5.1.4 below introduces. Using this approach a grid search for an optimal regularisation parameter (λ which was introduced earlier in section 3.3.2 in Chapter 3) is performed. For a random set of values for λ, cross validation was performed using part of the dataset as a validation set. Once the optimal regularisation parameter had been found from the set of regularisation parameters, the network was evaluated on an unseen section of the dataset defined as the test set. The training, validation and test sets were then shuffled for a new set of regularisation parameters and the process repeated. Once the

optimal λ was obtained from the many possible sets, the network's performance on a completely unseen test set was calculated.

The implementation of the R^{2}SP approach involved the creation of a new configuration file which defined the properties of both $stat_1$ and $stat_2$ (for example, the size of each layer and from which distribution and range their input weights were drawn) and created the two additional layers. As well as defining these new parameters, some standard scripts were updated to incorporate the activations and weights of the newly added layers. As this was implemented in the 1.0 version of the toolbox, this approach was quite long-winded; later versions such as RCToolbox version 2.0 and the newly released Python version, Oger [3], are much more flexible and allow layers to be added with ease, altering only the configuration file, rather than some of the core scripts. However, as these versions were not released at the time of implementation, all simulations were performed using version 1.0. Although some scripts were changed, the overall flow of a network simulation remained unchanged. The RCToolbox scripts that were updated are outlined in Table 4.1.

[3]Available from `http://organic.elis.ugent.be/organic/engine`

Table 4.1: The major changes made to the existing RCToolbox scripts to accommodate the added layers $stat_1$ and $stat_2$. The numbers in brackets after the script name indicate where these scripts were called during the simulation of a reservoir detailed in Figure 4.3 above. A brief description of the functionality of each script is also given.

Script name	**Brief description**	**Changes made**
dataset generation (1)	Creates/loads the dataset(s) to be used in the simulation and returns them in a Matlab structure.	To create two extra Matlab sub-structures where the activations of $stat_1$ and $stat_2$ will be stored.
rc_simulate_job (2)	Creates reservoir and calls simulation and training scripts.	To create $stat_1$ and $stat_2$ and to call new versions of simulation script (generic_simulate) and training script (cross_validate_grid).
generic_simulate (3)	Simulates the reservoir neurons for every timestep in the dataset.	To calculate the activations of the newly added neurons in $stat_1$ and $stat_2$.
cross_validate (5)	Perform cross-validation.	To call amended output weight training script (train_readout).

Continued on Next Page...

Table 4.1 – Continued

Script name	**Brief description**	**Changes made**
cross_validate_grid (6)	Perform cross-validation with a grid search for optimal regularisation parameter.	To call amended output weight training script (train_readout).
train_readout (4)	Trains output weights according to train function (ridge regression, for example).	To call newly updated script that creates matrix containing the R^{2}SP-reservoir, $stat_1$ and $stat_2$ activations and target outputs (construct_AB) and the script which assigns output weights to the relevant layer (split_up_w).
construct_AB (4)	Creates and returns a matrix containing R^{2}SP-reservoir, $stat_1$ and $stat_2$ activations and target outputs.	Concatenates activations of all three layers into one matrix.

Continued on Next Page...

Table 4.1 – Continued

Script name	**Brief description**	**Changes made**
split_up_w (4)	Assigns output weights to relevant layer. For example, output weights of reservoir are assigned to lin_readout.w.res2out.	Now assigns output weights of $stat_1$ and $stat_2$ to relevant variables.
apply_lin_readout (4)	Calculates the output of the network using the activation functions and the output weights of the network.	Uses activations and output weights from $stat_1$ and $stat_2$, as well as the R^{2}SP-reservoir.

A useful function of the RCToolbox is the ability to apply Fisher relabelling to a dataset [210]. Fisher labelling generally improves results for single or multi-class classification of datasets which contain an unbalanced amount of one or multiple classes, for example in a two-class problem where more instances of class 1 exist than class 2 or vice versa. In two-class problems, as a result of using a Least Squares approach to training a reservoir, the hyperplane which separates the two classes is closer to the class centre with more instances, in this example, class one. Using Fisher re-labelling, the under-represented class labels are changed from the interval $[-1, +1]$ to $[\frac{n_1+n_2}{n_1}, \frac{n_1+n_2}{n2}]$ where n_1 and n_2 are the number of examples of class 1 and 2 respectively. Figure 4.4 shows the effect of Fisher relabelling on the hyperplane separating two classes. In Verstraeten [210] Fisher relabelling was found to improve performance using the same reservoir by 1% on the spoken digit recognition task (introduced later in Chapter 5).

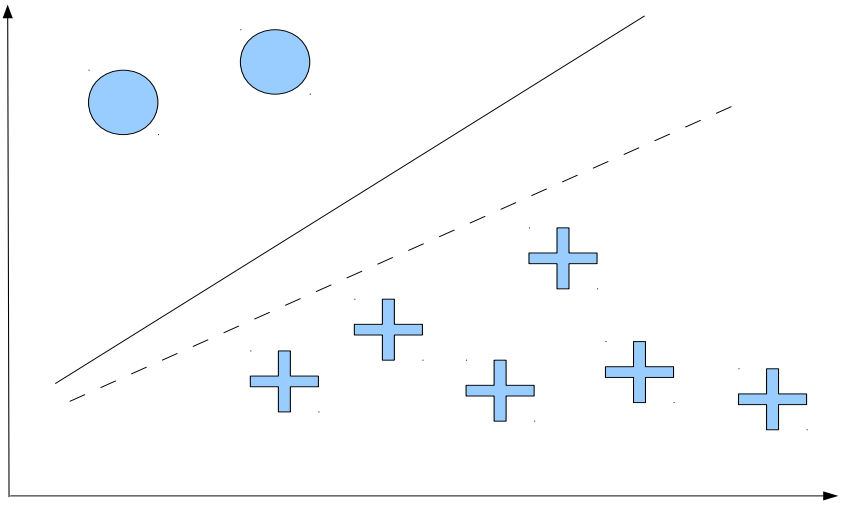


Figure 4.4: A schematic diagram illustrating the effect of Fisher relabelling on a hyperplane separating instances belonging to two classes. Originally, as a result of Least Squares training the hyperplane (dotted line) was closer towards the centre of the class with more instances. Fisher relabelling re-weights the class labels to move the hyperplane towards the centre between the two classes (solid line) (reproduced from Verstraeten [210]).

4.4.5 Comparative techniques

In order to validate the performance of the ESN, R^{2}SP and TD-ELM on the datasets described in the next chapter they were compared to a number of alternative machine learning techniques including Elman networks and time-delayed approaches, as well as techniques reported in other studies such as Hidden Markov Models (HMMs) and LSMs. For some of the tasks the Elman (EN), layer recurrent (LRN) and distributed time delay (DTDNN) ANNs were used as benchmarks and were compared against both ESNs, the novel R^{2}SP architecture and TD-ELM presented above.

An Elman Network (EN) [317] is a three layer neural network with a single hidden layer that provides feedback from its hidden layer activations to the input neurons through *context neurons* that contain unit delays. This allows the network to possess memory of its previous inputs through a history of hidden layer activations. Essentially, it resembles a one hidden layer multi-layer perceptron but with the addition of feedback. The Layer Recurrent Network (LRN) is an extension of the EN as it allows for any number of hidden layers and multiple transfer functions within each layer. Each layer also has feedback, as with the original EN, except for the last hidden layer. Figure 4.5 shows the two architectures.

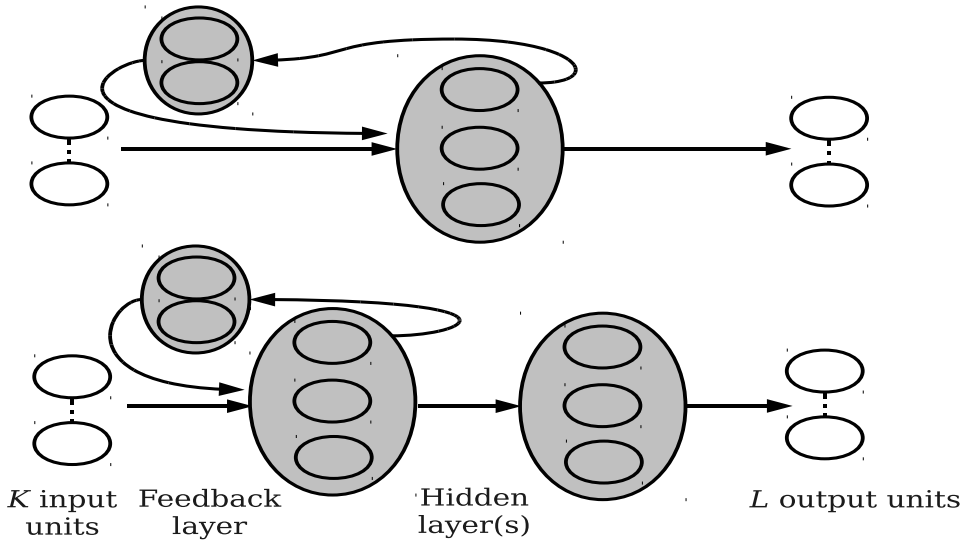


Figure 4.5: An Elman Network architecture (top) and the extended Layer Recurrent Network (bottom) with two hidden layers. Arrows show how activations are fed through weighted connections between layers. All weighted connections are modified during training here.

A Distributed Time-Delay Neural Network (DTDNN) [318] is a RANN which has a tapped delay line at the beginning of every hidden layer. This is an extension of the Focussed Time-Delay Neural Network (FTDNN) which consists of a feedforward

neural network with an additional tapped delay line present in the input layer. A tapped delay line presents input data at various time steps to a network, thereby allowing the network to contain a memory of the input which is equal to the length of the delay line. Figure 4.6 is a schematic representation of a two hidden layer DTDNN.

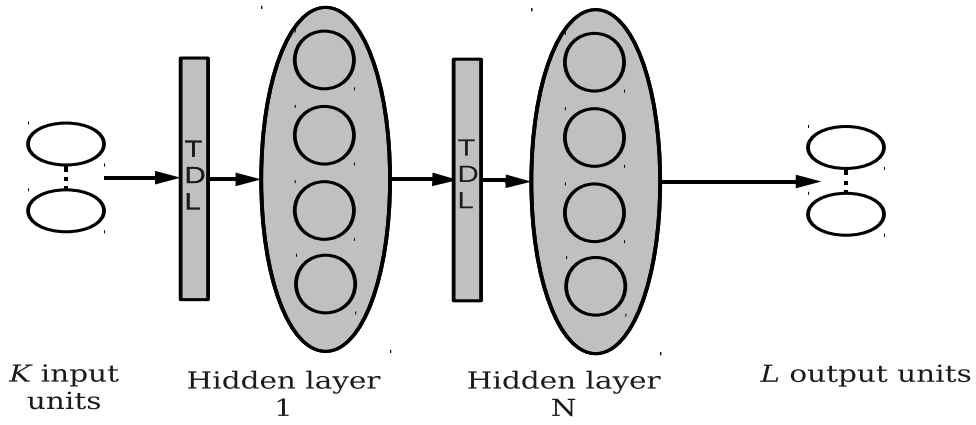


Figure 4.6: A two layer DTDNN which has a delay line (TDL) present at each layer. All weighted connections are modified during training as indicated by the sold lines.

All three of these networks were trained using backpropagation with an adaptive learning rate (Matlab *traingdx* function). This algorithm uses both momentum and adaptive learning rate terms in order to reduce the error of the network by calculating derivatives of the network's performance with respect to the weight and bias variables. Each variable is adjusted according to gradient descent with momentum. After each epoch of training, the learning rate is adjusted according to the increase or decrease in error. If the error decreases, the learning rate is increased by a factor of 1.05; if on the other hand the error increases by more than a factor of 1.04, the learning rate

is decreased by a factor of 0.7 and the update that led to the increase in error is not made. If the error increases by less than 1.04 then the learning rate remains the same for the next epoch. This allows a good learning rate to be used throughout training, rather than fixing the learning rate at network creation.

In addition to training the three RANNs using backpropagation with an adaptive learning rate as discussed above, an Elman network's performance when trained with backpropagation through time (BPTT) [319, 49] and real-time recurrent learning (RTRL) [223] was also investigated[4].

Backpropagation through time (BPTT) is an extended version of the standard backpropagation algorithm which was created to train RANNs (it was first described by Werbos [319] and was rediscovered by Rumelhart et al [49]). BPTT essentially works by 'unrolling' a network into a feedforward ANN which has T copies, one copy for every time step. This unrolling allows the error gradient of the network to be backprogated through time, as the local gradient at the current time step is influenced by the gradient from previous time steps. During training, the hidden layer neurons' activations are updated for each copy of the network, after which all of the network weights are updated. Figure 4.7 shows a network with three hidden neurons unrolled over time using the BPTT training algorithm. BPTT can be computationally expensive, as the activations of each hidden layer neuron at each time step require saving, which can become problematic for networks with a large number of hidden layer neurons. When presented with an infinite input sequence, the computational requirements are given by $\theta((m+n)L)$, where θ is the exact order of magnitude of computational complexity, L is the total length of time for which the algorithm is executed, m is the dimension of

[4]All of these training algorithms were adopted from [320]

the input and n is the number of neurons [321]. As BPTT is a gradient descent-based approach, it also suffers from problems such as local minima, slow convergence and vanishing gradients as were discussed previously in Chapters 1 and 3. The memory capacity of BPTT trained networks is often quoted as short; a network may typically struggle to recall inputs longer than 20 time steps in the past, as a result of the fading backpropagated error gradient information [182]. For a thorough explanation of the BPTT algorithm see Haykin [40].

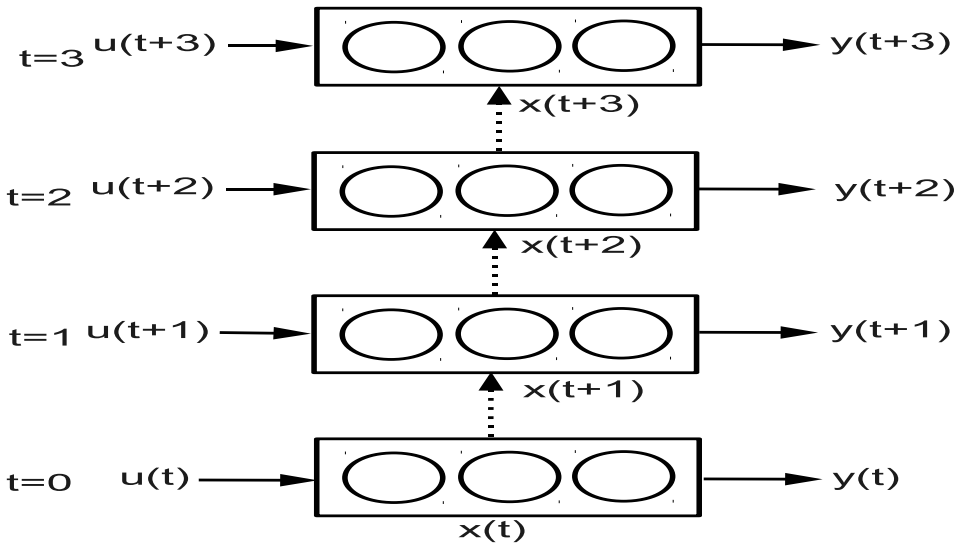


Figure 4.7: A RANN containing three hidden layer neurons unrolled over time using the BPTT training algorithm. The dashed arrows show how the activations of the hidden layer neurons at time t are determined from the hidden layer activations at the previous time step, $t-1$, and the current input, $\mathbf{u}(t)$.

Real-time recurrent learning (RTRL) [223] adjusts the weights of the network over the time course of the input data, hence the term real-time. RTRL can be seen as a converse approach to BPTT as it propagates the error forward, rather than backwards, in time [321]. RTRL works by calculating the sensitivity of the output of a neuron at a time step within the dataset to a small increase in its weight, also known as the error gradient. This takes the effect of changing the weight over the neuron's trajectory from the start of the dataset, i.e. t_0, to the current time step t, into account. The initial state of the network, the input between t_0 and t and the other weights of the network are assumed to be unchanged [223]. The sensitivity of each neuron is calculated for each time step which is influenced by the activations in the network at that time as well as the neuron's previous activation values. Comparing this sensitivity to the error at time t results in the error gradient which, as it is computed at time t, is available in real-time.

4.5 Research hypotheses for research question 2 and contribution 2

The hypotheses relating to research question 2 and contribution 2 (outlined earlier in section 1.5 of Chapter 1) are outlined in Table 4.2 below. This table shows which architectures that the R^2SP was compared against for several datasets using a specific error measure. The hypothesis was that the R^2SP would overcome the trade-off between non-linear separation and short-term memory, thus improving performance.

Table 4.2: The different architectures R^{2}SP's performance was compared against when applied to various datasets. Here the hypothesis is that the R^{2}SP will offer a significant improvement in performance.

Architecture R^{2}SP outperforms	**Dataset**	**Error measure**
ESN, TD-ELM, EN, LRN, DTDNN, EN-BPTT, EN-RTRL	4^{th} order polynomial (see section 5.1.1)	NRMSE
ESN	Extended polynomial (see section 5.1.2)	NRMSE
ESN	Extended delayed XOR (see section 5.1.3)	NRMSE
ESN	Spoken digit task: trained using noisy utterances (see section 5.1.4)	WER
ESN, TD-ELM, EN, LRN, DTDNN, LSM, TRM, CRBM, HMM, OSBP-res, RLS-res, LSTM	Spoken digit task: trained using clean utterances (see section 5.1.4)	WER

Continued on Next Page...

Table 4.2 – Continued

Architecture R^2SP outperforms	**Dataset**	**Error measure**
ESN, TD-ELM	Laboratory controlled mesh (see section 5.2.1)	% defects incorrect (E_{ex_loss}), Sensitivity, Specificity, PPV, NPV, MCC, % specific defects detected
ESN, TD-ELM	Temple Moor (see section 5.2.2)	% defects incorrect (E_{ex_loss}), Sensitivity, Specificity, PPV, NPV, MCC

Inspection of Table 4.2 shows that the R^2SP's performance was compared against an ESN for every dataset investigated. This is a result of the first research question (outlined in section 1.5 of Chapter 1) which focuses on whether or not the trade-off between non-linear separation and short-term memory present within an ESN when applied to highly non-linear time-series datasets can be overcome. As the TD-ELM was an alternative approach to the R^2SP in overcoming this trade-off, it was also investigated for the majority of the datasets, with the exception of the extended polynomial and extended XOR tasks. The omission of the TD-ELM's performance when applied to these datasets was due to time restrictions, and is an area for future work. The traditional RANNs (EN, LRN, DTDNN, EN-BPTT and EN-RTRL) were only applied to the fourth order polynomial and the spoken digit task which was due to their slow

training times. As they also offered the worst performance (see Chapter 6 for more details), they were not applied to the more complex and larger datasets where poor performance as well as longer training times were also expected. Inspection of the number of different comparative architectures when applied to the spoken digit task when trained using clean utterances reveals that many architectures were compared here. This is due to the wide literature available on different architectures' performances when applied to this dataset, and hence, are included for comparison. Given the time available for this project, the performance of these architectures when applied to the remaining datasets was not investigated and is left for future work.

4.6 Summary

This chapter has outlined two potential solutions to the trade-off present between a reservoir's ability to recall past inputs and to perform non-linear separation of highly non-linear data. The first proposed approach, Reservoir with Random Static Projections (R^2SP) is a novel RC architecture presented for the first time here and borrows from the principles of the Extreme Learning Machine (ELM), where, as with RC, only the output weights are altered during training. The ELM has no inter-neuron connections in its hidden layer and has no memory of previous inputs. The task of the ELM layers is to perform instantaneous non-linear transformations of their input data onto a higher dimensional state-space. Two ELM layers are added in the R^2SP architecture, one in parallel with its R^2SP-reservoir to provide an instantaneous non-linear transformation of the input ($stat_1$), while the second ELM layer ($stat_2$) receives input from the R^2SP-reservoir, performing another instantaneous non-linear transformation this time of the R^2SP-reservoir states at each time step.

The addition of these two layers allows the R^{2}SP-reservoir to be tuned towards a higher memory capacity, as the non-linear transformation of the data is mainly taken care of by $stat_1$, which also reacts instantly to any change in input data. The second ELM layer, $stat_2$ will facilitate the readout by transforming the R^{2}SP-reservoir activations onto a higher dimensional state space.

In the second approach, TD-ELM, adopts a similar idea without the use of a reservoir to provide short-term memory. The short term memory here is provided by a sliding window of the input data, where each time-step contained within the sliding window is presented as an input to the feedforward hidden layer of an ELM. The non-linear transformation of the input data is performed by the memoryless feedforward ELM hidden layer, whose size is determined using an empirical search. This approach has been proposed in several studies previously, where in one study [311] it was shown to outperform an ESN whilst offering faster training times when applied to several non-linear datasets.

The separation of non-linear mapping and short-term memory may allow these two networks to offer improved performance when compared to an ESN when applied to tasks which require high amounts of both. Candidate examples of such tasks are introduced in the next chapter, with the performance of standard RC, R^{2}SP, TD-ELM and other benchmark approaches when applied to these example datasets explored in Chapter 6.

Chapter 5

Case studies

This chapter presents the scenarios and datasets that were used in this research to investigate the properties and performance of the R^{2}SP and TD-ELM architectures outlined in the previous chapter. The performance of these two architectures and the performance of an ESN were investigated using artificially created datasets and real-world data from the speech recognition domain. Furthermore, case studies containing data collected using the EMAD from a laboratory controlled reinforced concrete corrosion experiment and from a real-world on-site location were analysed. The performance of an ESN, R^{2}SP and TD-ELM when applied to these datasets is also compared against several other ANN architectures below.

5.1 Non-concrete benchmark case studies

This section outlines the synthetic and real-world datasets from domains outside the area of defect detection within reinforced concrete that were used in this research to give greater insight into the dynamics of the reservoirs studied. Although some of these datasets are artificial, they were extremely useful for exploring the internal dynamics

of the different reservoirs used.

5.1.1 Fourth order polynomial

This dataset contained highly non-linear time-series data which also required some short term memory. It consisted of one input and one output value and the networks were given the task of replicating the polynomial's output when driven by random input, essentially outputting a non-linear delayed version of the input data. The input data, $\mathbf{u}$, was randomly drawn from a uniform distribution between -1 and +1, whereas the target output, $\mathbf{y}_{tgt}$, was determined by the following polynomial:

$$\begin{aligned}\mathbf{y}_{tgt}(t) = \quad & c_{00} + c_{10}\mathbf{u}(t) + c_{01}\mathbf{u}(t-1) + c_{20}\mathbf{u}(t)^2 + c_{02}\mathbf{u}(t-1)^2 + c_{11}\mathbf{u}(t)\mathbf{u}(t-1) + \\ & c_{30}\mathbf{u}(t)^3 + c_{21}\mathbf{u}(t)^2\mathbf{u}(t-1) + c_{12}\mathbf{u}(t)\mathbf{u}(t-1)^2 + c_{03}\mathbf{u}(t-1)^3 + c_{40}\mathbf{u}(t)^4 + \\ & c_{31}\mathbf{u}(t)^3\mathbf{u}(t-1) + c_{22}\mathbf{u}(t)^2\mathbf{u}(t-1)^2 + c_{13}\mathbf{u}(t)\mathbf{u}(t-1)^3 + c_{04}\mathbf{u}(t-1)^4\end{aligned} \tag{5.1}$$

where each c_{ij} is a polynomial coefficient randomly drawn from the same distribution as the input. Equation 5.1 can be more compactly written as follows:

$$\mathbf{y}_{tgt}(t) = \sum_{i=0}^{4}\sum_{j=0}^{4-i} c_{ij}\mathbf{u}^i(t)\mathbf{u}^j(t-1) \text{ s.t. } i+j \leq 4 \tag{5.2}$$

where c_{ij} denotes the same polynomial coefficients as shown in equation 5.1 above.

Inspection of the polynomial's output reveals that the dataset requires mostly non-linear separation as a result of the high order of the polynomial. However, some short-term memory is also required as the current output is influenced by the previous input, $\mathbf{u}(t-1)$, as well as the current input at time t. Figure 5.1 shows an example of

the random input and polynomial target output over 100 timesteps. The performance of the networks studied was calculated using the Normalised Root Mean Squared Error (NRMSE):

$$NRMSE = \sqrt{\frac{\langle \left(\hat{\mathbf{y}}(t) - \mathbf{y}_{tgt}(t)\right)^2 \rangle_{t_{tot}}}{\sigma^2 \mathbf{y}_{tgt}}} \tag{5.3}$$

where σ_y^2 is the variance of the target output, $\langle\rangle_{t_{tot}}$ is the average over the whole time series, t_{tot} is the total number of patterns, $\mathbf{y}_{tgt}$ is the target output and $\hat{\mathbf{y}}(t)$ is the actual output from the network.

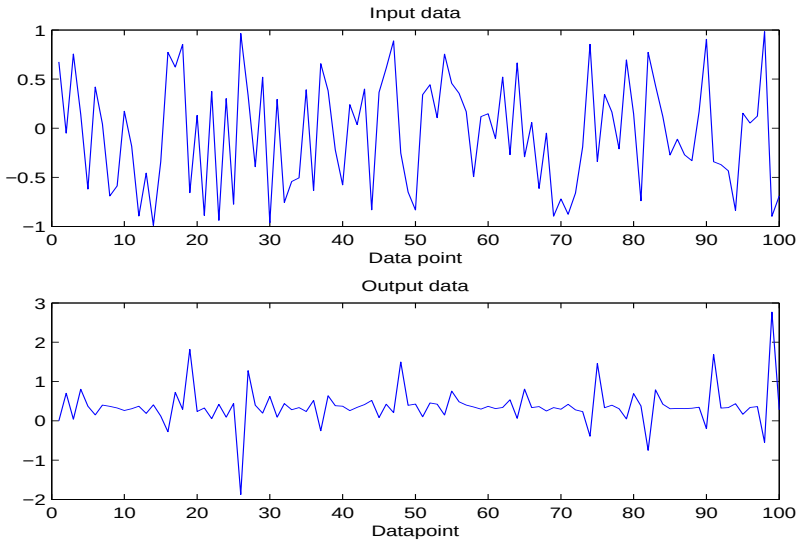


Figure 5.1: An example of random input and the 4^{th} order polynomial target output over 100 timesteps.

5.1.2 Extended polynomial

The fourth order polynomial presented in section 5.1.1 was also extended to allow the order and delay of the polynomial to be varied. An increase in the order of the

polynomial increases the non-linearity of the dataset, while an increase in the delay increases the memory required from a network. The input data was selected from the same range as the original fourth order polynomial above. The extended polynomial's output is given by:

$$\mathbf{y}_{tgt}(t) = \sum_{i=0}^{p} \sum_{j=0}^{p-i} c_{ij} \mathbf{u}^i(t) \mathbf{u}^j(t-d) \text{ s.t. } i + j \leq p \tag{5.4}$$

where d is the delay, p is the power, or order, of the polynomial and all of the other terms are as described for equation 5.2. As more coefficients are required (since the power of the polynomial was planned to be increased up to 150), the first 15 coefficients used in the extended polynomial are not the same as those used in the fourth order polynomial presented above. This extension allowed the non-linearity and memory contained in the output to be varied, where p controlled the order of the polynomial and the non-linearity of the dataset and d controlled the memory required from the network which would allow the properties of an ESN and the R^2SP architecture to be explored further. Increasing both of these parameters at the same time increases the difficulty of the task, although this task does not have the added non-linearity of the extended XOR task described in section 5.1.3 below. The performance of the network was evaluated using the NRSME outlined in equation 5.3. Figure 5.2 shows the target output data when p and d both have values of 5, 20, 50 and 150.

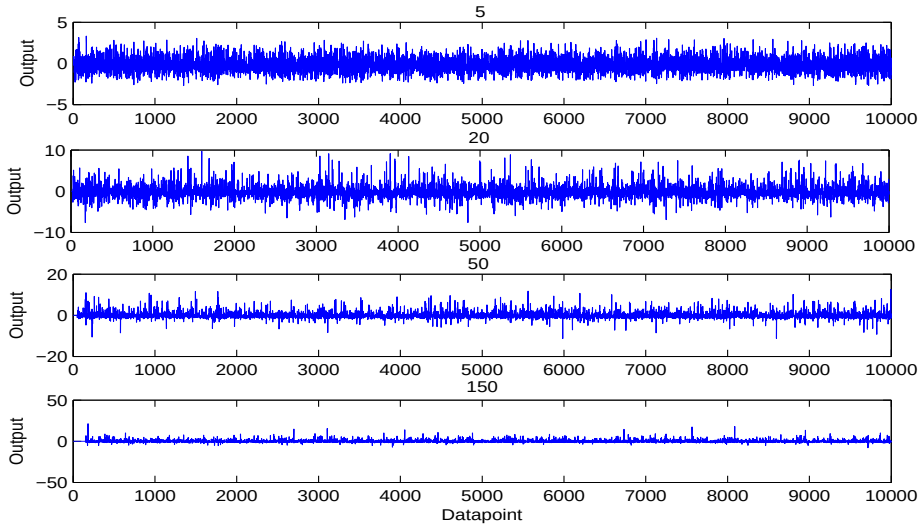


Figure 5.2: The target output of the extended polynomial task when p and d both have values equal to 5 (top plot), 20, 50 and 150 (lower plot).

5.1.3 Extension of delayed exclusive OR (XOR) task

This dataset is an extension of a delayed 3-bit XOR task outlined by Schrauwen et al [322] where the aim was for the network to give an output of the binary XOR of three delayed bits. This dataset, therefore, was highly non-linear and required some memory of previous inputs. The extended version of this dataset was investigated by Verstraeten et al [226] to give control over the amount of non-linearity and memory the dataset requires. This is an extended version of the classic XOR problem where the inputs are transposed to the continuous domain [-1,+1]. By taking the product of the two inputs, the same non-linear mapping of the outputs can be computed as shown in Table 5.1, exemplified by the input values +1 and -1.

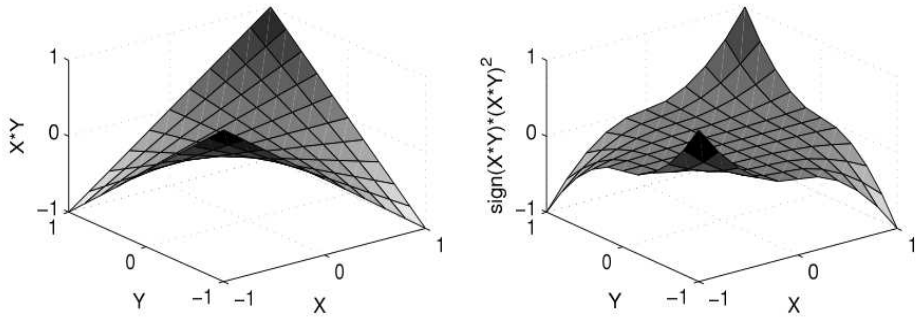
Table 5.1: The extended XOR task with continuous valued inputs calculated by taking the product of the two inputs.

Input	**-1**	**1**
-1	1	-1
1	-1	1

The actual input consisted of uniformly distributed random values between the range -0.8 to +0.8, essentially giving a white noise signal. A network was then required to give the following output at time t given a delay, d, and a power, p:

$$\mathbf{y}_{tgt}(t) = sign\left(r(t-d)\right) abs\left(r(t-d)\right)^p \tag{5.5}$$

where r is the product of two successive delayed inputs, $r(t-d) = \mathbf{u}(t-d)\mathbf{u}(t-d-1)$. As with the extended polynomial, the amount of memory required from the network for this task was determined by the delay parameter, d, while the amount of non-linearity required from the network is determined by the power parameter, p, giving explicit control over the amount of memory and non-linearity contained within the dataset. This was shown to be helpful by Verstraeten et al [226] in analysing these two main requirements of an ESN. The performance of the ESN was evaluated using the NRMSE previously described by Equation 5.3. This dataset contains added non-linearity in the output as a result of multiplying two timesteps to obtain $r(t-d)$ and then raising this to the power of the parameter p. Figure 5.3 shows the non-linearity obtained from the product of the extended XOR (left) and the increased non-linearity achieved by multiplying this by a power (in the figure it is raised to the power of two).



(a) Simple product of two variables.

(b) Square of the product of two variables.

Figure 5.3: The non-linearity created from taking the product of two values from the continuous domain (left) and raising it to a power (right) which increases the non-linearity further (reproduced from Verstraeten et al [226]).

By raising the product of the non-linearity of the target output by the power, the task becomes very difficult after only small increases in the power and delay parameters, which is shown from a plot of the error of an ESN, presented by Verstraeten et al [226], in Figure 5.4.

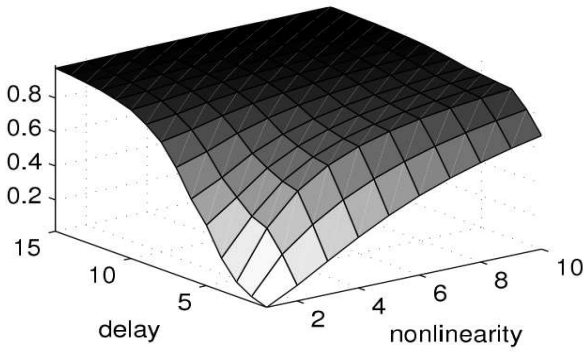


Figure 5.4: The error of an ESN as the delay and power of the extended XOR task increase. Notice the fast increase in error after small increases in the two parameters indicating the difficulty of the task (reproduced from Verstraeten et al [226]).

5.1.4 Spoken digit recognition

A subset of the TI46 isolated speech dataset [248] was used which contained 10 spoken utterances of 10 digits (zero to nine) each spoken by five different female speakers. This gave a total of 500 spoken utterances. In addition, noise in the form of recorded speech *babble noise*[1] from a cafeteria was added to the training and test data to make it more challenging to classify, using various signal-to-noise ratios (SNRs) (training a reservoir with noise-free data gave a very low error around 3% as shown in Chapter 6). This data was then preprocessed using the Lyon cochlea model [323] with a subsampling rate of 128, giving a 77-dimension time-varying feature vector (in other words, 77 input patterns). For each spoken digit the input data consisted of a $77 \times T$ matrix (where T is the length of the spoken utterance) and the target output data was a vector of size 10 (one target output for each utterance where a value of one for one utterance indicated it was the current input to the network and all other outputs were set to zero).

Previous work conducted by Verstraeten et al [210, 247] used this dataset and applied it to spiking neurons, namely LIF neurons which were introduced earlier in section 3.2. It was also found from a number of preprocessing models such as the Lyon cochlea model [323], the Seneff cochlea model [324] and the Mel-Frequency Cepstral Coefficients (MFCC) technique, which is often used as the preprocessing technique for HMMs [325] (see Verstraeten [210] for a good overview of the different models), that the Lyon cochlea model gave the smallest error of 8.6%. The Lyon cochlea model is essentially a model of the human inner ear (cochlea) which contains a filter bank that is selective to different frequencies (like the human cochlea). The model consists of half-wave

[1]This is from the NOISEX database which can be downloaded free from Rice University (`http://spib.rice.edu/spib/data/signals/noise/babble.html`).

rectifiers and adaptive gain controllers which model the response of the cochlea hair cells to the input sound to give an output in the form of a cochleagram. A cochleagram contains coefficients which indicate the probability of a hair cell firing when presented with the input sound, as is shown by the right-most image in Figure 5.5. This model is more suited to preprocessing speech data prior to input to a reservoir than the MFCC approach as the MFCC technique is not well-suited to the way reservoirs process spatio-temporal data as a result of it being created especially for HMMs in order to overcome some of the problems associated with them [249]. The network's performance was measured using Word Error Rate (WER) which is the fraction of incorrectly classified words:

$$WER = \frac{N_{nc}}{N_{tot}} \tag{5.6}$$

where N_{nc} is the number of incorrectly classified words and N_{tot} is the total number of words presented to the network [210].

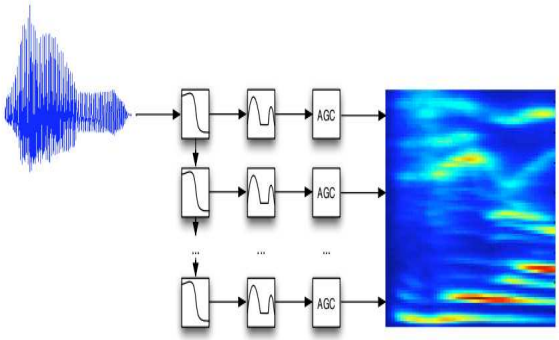


Figure 5.5: An overview of the Lyon cochlea model used to preprocess the TI46 dataset showing a raw spoken digit (left), a group of filter banks (left-most plots centre), half-wave rectifiers (centre plot centre) and adaptive gain controllers (right-most plot centre) which produce a cochleagram (right plot) (reproduced from Verstraeten [210]).

5.2 Defect detection within reinforced concrete

As was introduced in Chapter 1, one of the main applications and, hence, major motivations behind this research was the automated detection of defects in EMAD data collected from scanning reinforced concrete structures. This would speed up the time involved in data analysis which is currently performed using a visual procedure, making the process cheaper while keeping a similar level of accuracy. This could allow, in turn, for more assessments to be conducted, improving the overall condition of our infrastructure.

5.2.1 Laboratory controlled reinforcing mesh

5.2.1.1 Introduction

An experiment using a standard steel reinforcing mesh was designed in order to introduce and capture data from typical defects found within real-world structures. Defects were inserted into a steel concrete-reinforcing mesh prior to concrete encasement to provide typical defect signatures for data capture using the EMAD technique. Corrosion of the encased steel mesh was then accelerated by immersing it in an aggressive environment using a sodium chloride solution. This meant that the introduced defects could be monitored over time as they deteriorated. Any new defects which resulted from the corrosive environment could also be detected and monitored.

This approach has several advantages over other approaches widely reported in the literature, such as simulations which use computer models of deteriorating structures to provide data for analysis [326, 327, 328, 329]. In contrast this study uses real defects that have been monitored and their EMAD data collected over a 12 month period from the creation of the experiment to the end of the research project. One disadvantage of

using an experimental approach in this case is that defects take time to develop and, in some cases, this can be longer than the time-scale of a research project, necessitating the use of corrosion acceleration measures (see section 5.2.1.3 below).

This laboratory experiment provided typical EMAD signatures of different mesh defects over a period of time, something which has not been reported in the literature before. These signatures were used for the training and testing of neural networks to see if the networks could be trained to detect, and possibly distinguish between, the different types of defect based on their EMAD signatures. Once trained, unseen data collected from real-world structures could be presented to the networks which, based on what they have learned, will be used to classify the new, unseen data. This approach overcomes the problem of model inaccuracies when trying to simulate the formation and development of defects within steel. Collecting EMAD data in this manner also meant that the defect signatures captured from this mesh could be used to train ANN classifiers to detect defects in other mesh-like structures with little or no retraining. The proposed approach has the potential to enable a fast and accurate structural assessment to be made, reducing costs and improving the service life of structures worldwide.

5.2.1.2 The experimental reinforcing mesh

The reinforcing mesh was 2.2 m wide by 4.6 m long and consisted of a series of reinforcing bars (rebars) arranged longitudinally and transversely every 19.5 cm. The steel rebars were 5 mm thick. In total, the mesh contains 11 rebars lying longitudinally and 23 rebars lying transversely. In order to use supervised neural networks to learn and automatically detect unseen defects, training data must be presented to them with target outputs. For example, EMAD data which represents an area of corrosion needs

to be labelled with target outputs so that the network can learn to recognise the EMAD input data as corrosion and thereafter classify similar unseen input data patterns as corrosion also.

Hitherto, a major problem with using data collected from real-world structures has been determining what ground truth conditions lie beneath the surface of the concrete structure. Typically, investigations into the condition of the steel buried beneath the concrete are often carried out at a later date by another contractor, or not at all. Even when concrete is removed and the rebars are inspected, the actual condition of the steel in the concrete can never be accurately captured as some of the material is destroyed during the removal of the concrete around the steel. Using a reinforcing mesh under controlled conditions enabled the creation and collection of a dataset recording the evolution of typical defects which can then be used to train neural networks. Figure 5.6 shows the configuration of the mesh. Note that two sides (left and bottom) have 'open cells' where rebars protrude out from the mesh, while the remaining two sides (right and top) have 'closed cells' with no protruding rebars.

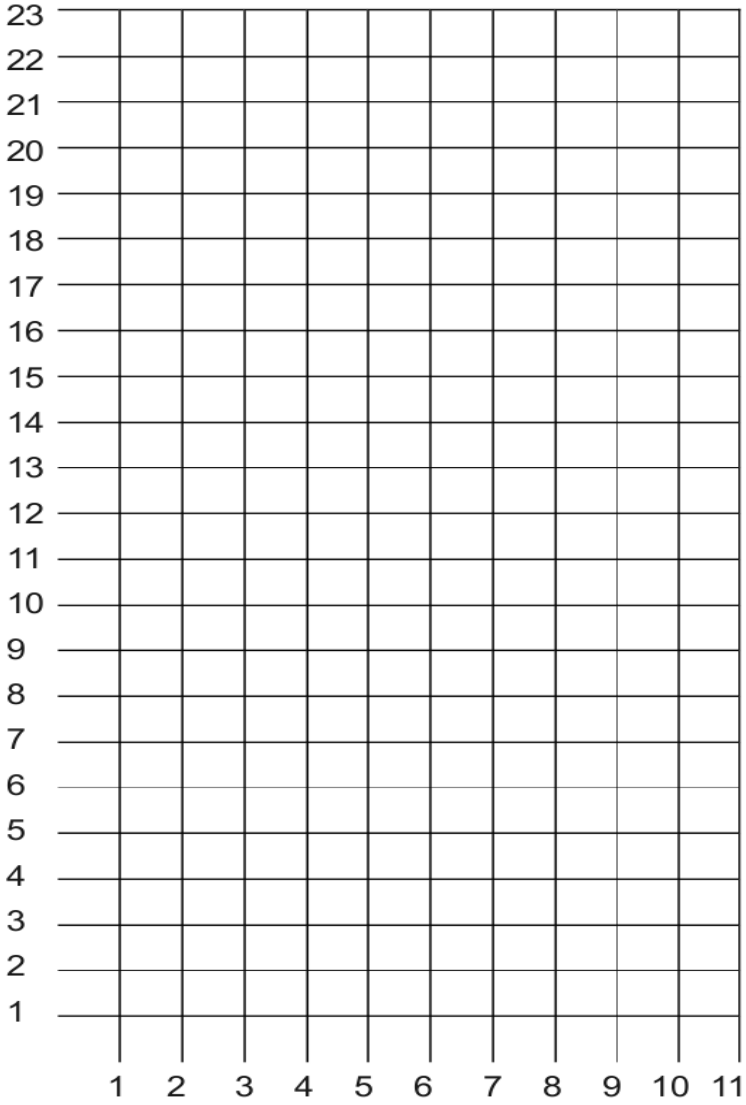


Figure 5.6: A schematic representation of the reinforcing mesh with each of the rebars numbered. Note the open cells to the left and bottom, and the closed cells to the right and top (not to scale).

A total of 12 defects of three broad types were introduced into the rebars of the reinforcing mesh, consisting of corrosion defects, break defects within a rebar, or a combination of the two. Breaks consisted of two types: a break on a straight section of rebar and a break on an intersection of rebars. Six defects were created towards the centre of the mesh, while the remaining six were created towards the edges of the mesh. The reason defects were created within these two regions is because the remanent fringing field towards the ends of the bars in the mesh is different from the fringing field near the middle of the mesh. For this reason two areas of the mesh were labelled as 'end effect' zone and 'normal' zone. The EMAD process of energisation creates larger signals at the area to which it was applied last. If, for example, a mesh were to be energised from left to right, bottom to top, a larger end effect would be observed at the top right corner of the mesh when compared to the bottom left of the mesh. This is expected from the magnetic principles of EMAD operation, but to a lay observer, and a neural network, this could lead to misclassification of areas which appear to be defect areas, but are in fact magnetic end-effects. This phenomenon has also been reported in separate studies [20, 330]. Figure 5.7 shows the effect of applying a magnetic field to a sheet of steel, where the dark and light areas show a negative and positive strength magnetic field respectively. Note that although the dark area is negative, its strength is weaker compared to the positive strength of the magnetic field as shown by the scale on the right.

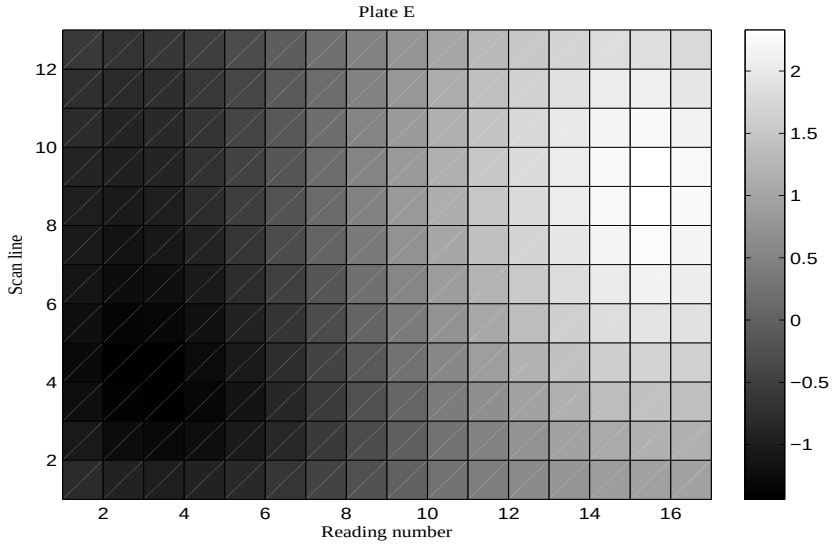


Figure 5.7: A plot showing the magnetic fringing field from a steel plate when energised. The energisation process was applied from the bottom left to the top right of the plate. The lighter area indicates a stronger positive magnetic field, whereas the dark area indicates a comparatively weaker negative magnetic field (from Sherratt [330]).

In order to gain maximal insight into the condition of the mesh over time, the experimental design required that it was scanned along the rebars and in between the rebars in both longitudinal and transverse directions. Scans offset at 30°, 45° and 60° were also regularly taken. Plywood boards were placed over the top of the mesh with all of the above directions marked to aid data collection and to ensure the consistency of the data acquisition throughout the experiment. Table 5.2 shows an overview of the 12 defects introduced, along with their unique identities and their approximate location within the collected data when scanning longitudinally and transversely. Based upon the ground truth position of the defects within the mesh, the locations of defect signatures in the data collected were accurately determined when labelling training

and testing data by dividing a defect's approximate spatial location by the sampling rate of the EMAD (the EMAD takes a reading every 6.3 mm). For example, defect 1C, which is located at rebars (2,21) where 2 is the 2^{nd} longitudinal rebar and 21 is the 21^{st} transverse rebar, should occur 420 cm into the mesh when scanning longitudinally. Dividing this by the distance between EMAD samples therefore gives the approximate datapoint at which the defect should occur. Using defect 1C as an example again, this would give the equation, $dataPoint = 4200/6.3 = 667$, meaning that defect 1C should occur close to the datapoint 667 when scanning longitudinally.

Table 5.2: The 12 different types of mesh defect which were introduced prior to concrete encasement.

Defect ID	**Area**	**Type**	**Longitudinal Location** (datapoint)	**Transverse Location** (datapoint)
1C	End effect zone	Corrosion on intersection of rebar	667	63
2C	Normal	Corrosion on intersection of rebar	127	127
3C	Normal	Corrosion on straight section of rebar	270	127
4B	Normal	Break on straight section of rebar	365	159
5B	Normal	Break on intersection of rebar	540	159
6BC	End effect zone	Break with corrosion on straight section of rebar	698	175
7BC	Normal	Break with corrosion on intersection of rebar	127	223
8BC	Normal	Break with corrosion on straight section of rebar	270	223
9B	End effect zone	Break on intersection of rebar	95	317
10C	End effect zone	Corrosion on straight section of rebar	302	317
11BC	End effect zone	Break with corrosion on intersection of rebar	508	317
12B	End effect zone	Break on straight section of rebar	651	317

5.2.1.3 The defects

As mentioned previously in section 5.2.1.1, 12 different types of defect were created in the steel mesh before it being set in concrete. This would allow the collection of EMAD data recording the corrosive time course of each defect. Any defects which appeared later during the experiment that looked similar to this signature could also be classified as this type of defect.

5.2.1.4 Introducing breaks into the mesh

Breaks were created using an angle grinder with a 2.5 mm blade. Figures 5.8 and 5.9 show examples of breaks in the middle of a rebar and at an intersection point.

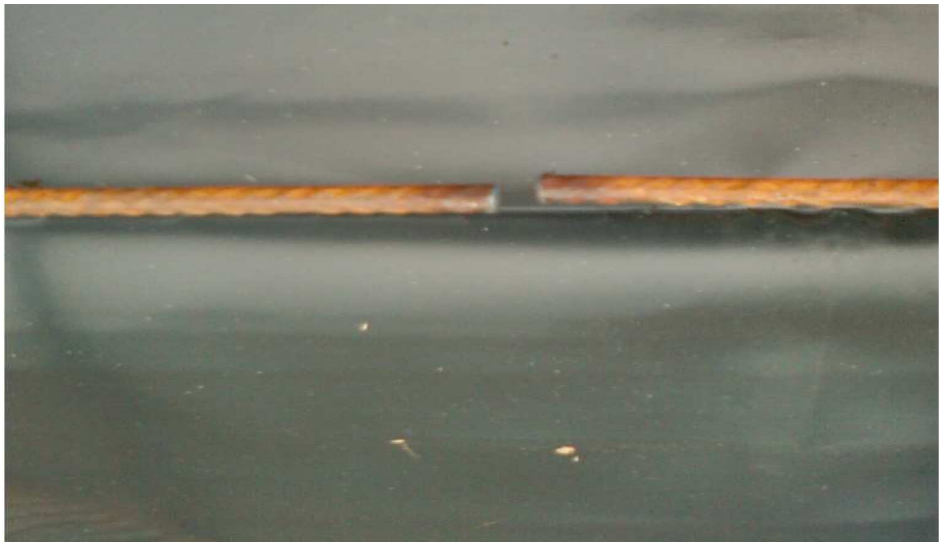


Figure 5.8: An example of a break inserted into the reinforcing mesh in the middle of a straight section of a rebar.

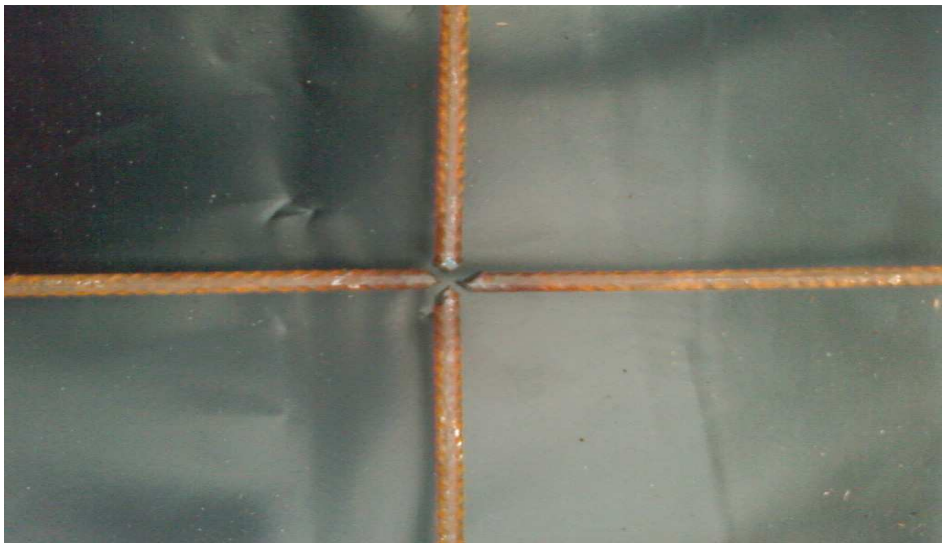


Figure 5.9: An example of a break inserted into the reinforcing mesh at an intersection of rebars.

Areas where breaks were to be inserted were chosen after some preliminary experimental testing. Firstly, a break was introduced into a mesh to assess the spatial extent over which this could be detected using the EMAD. This was to ensure that EMAD signatures from multiple defects would not be superimposed or interfere with each other. The investigative break was inserted on an intersection of rebars, at location (5,17). The data collected from energising and scanning longitudinally along rebars four, five and six before break insertion and after break insertion is shown in Figures 5.10, 5.11, 5.12, 5.13, 5.14 and 5.15.

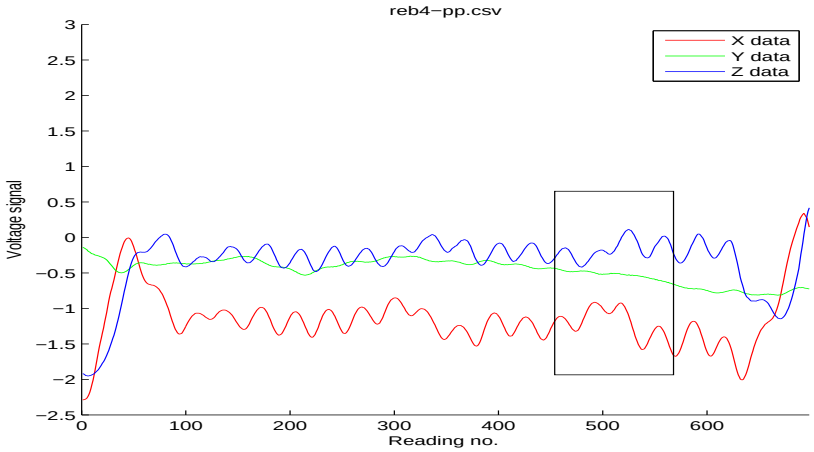


Figure 5.10: Data collected prior to break insertion along rebar four. The black box indicates the location of the break insertion in the scan line where a defect signature is expected to appear (see Figure 5.11 below).

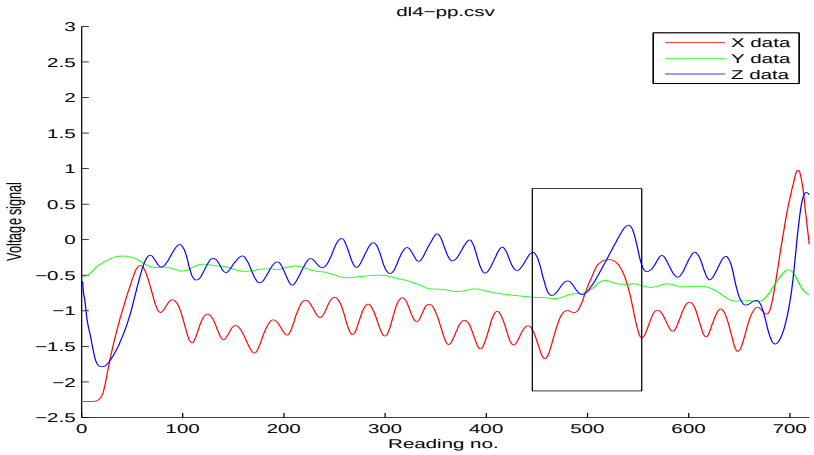


Figure 5.11: Data collected after break insertion along rebar four with the signature of the break highlighted by the black box.

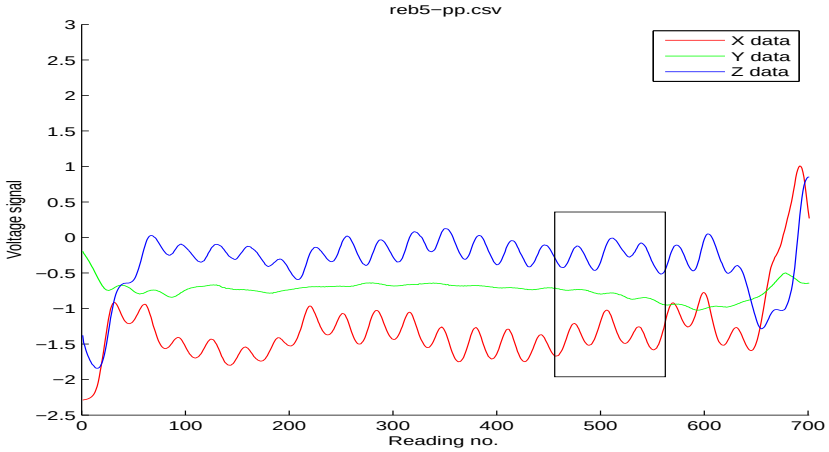


Figure 5.12: Data collected prior to break insertion along rebar five. The black box highlights the area where the signature of the break is expected to appear (see Figure 5.13 below).

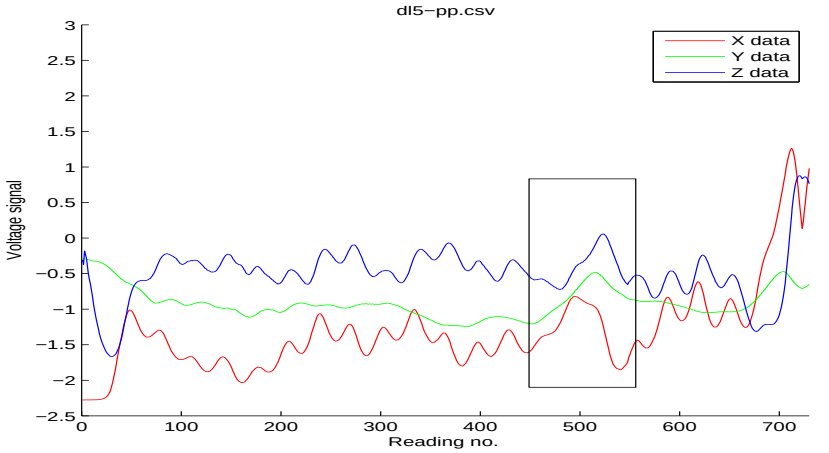


Figure 5.13: Data collected after break insertion along rebar five with the signature of the break highlighted by the black box.

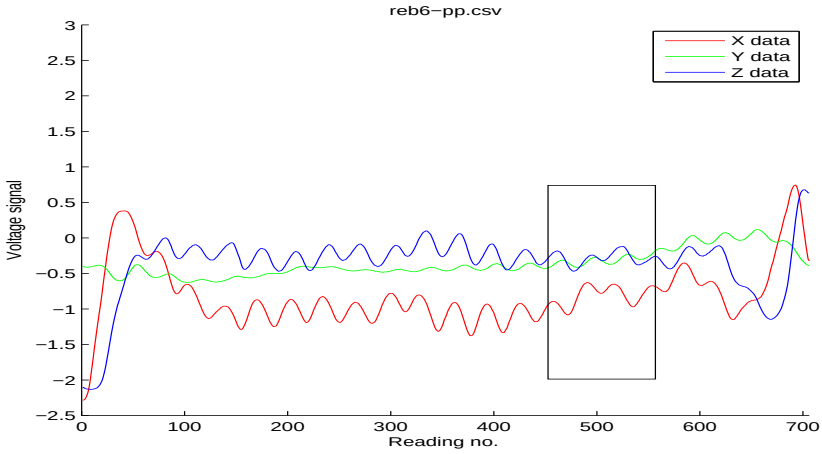


Figure 5.14: Data collected prior to break insertion along rebar six with the expected location of the defect signature highlighted by the black box (see Figure 5.15 below).

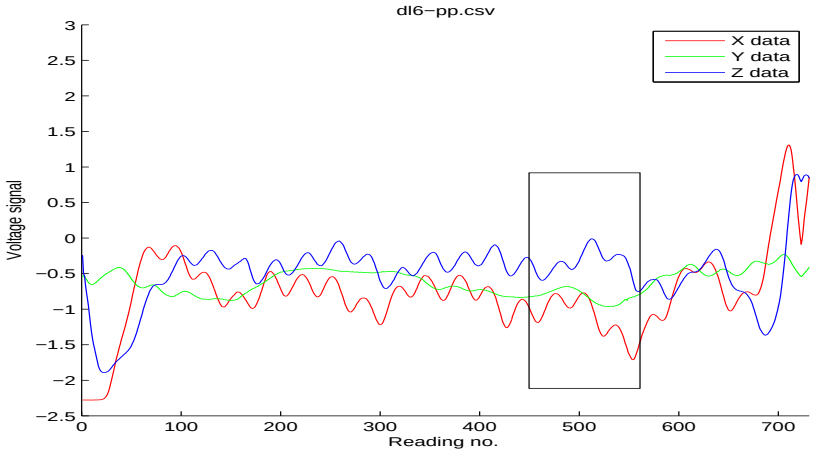


Figure 5.15: Data collected after break insertion along rebar six with the signature of the break highlighted by the black box.

Inspection of Figures 5.10, 5.12 and 5.14 shows the healthy mesh with a periodic ripple effect through each scan, especially in the X and Z direction plots. This effect is

caused by the probe passing through the small magnetic fringing fields created by the presence of each rebar. For example, Figure 5.10 contains 20 peaks in the X direction data, indicating 20 rebars. The remaining three rebar peaks were probably obscured by the big deviations at the start and end of the scan line caused by the larger fringing fields found at the edges of the mesh. Comparing these figures to Figures 5.11, 5.13 and 5.15 shows a change in all scans, roughly between 450^{th} and 550^{th} EMAD datapoint, especially in scan five which was expected as the probe ran directly over the break at this point. From these graphs the break's EMAD spatial extent can be determined transversely to be at least as far as one rebar, whilst longitudinally the break's EMAD signal extent is at most 100 data points, which as the probe takes a reading every 6.3 mm, equates to 63 cm.

In order to see how far the break could be detected transversely across the mesh, scan lines further away from the break were analysed where it was found that the effects of the break could be detected in scan lines three and seven, while it was no longer present in scan line two, and only slightly visible in scan line eight (for more details and the data collected for each scan line, see Appendix C). Once set in concrete, the distance between the probe and the mesh would be greater as a result of the layer of embedding concrete as well as the added distance between the plywood boards and the probe. Based on this, an exclusion zone of one and a half rebars square around each introduced defect was determined as sufficient. The type of defect used for the above test was at an intersection of rebars, which creates the biggest magnetic signal from the mesh as a total of four co-located rebars all have magnetic fringing fields which leak from the mesh at such an intersection. Therefore, any other type of defect would give a much smaller defect signal. Since the introduced defects would deteriorate over time creating larger signatures (as a result of the corrosion acceleration process explained

in section 5.2.1.5 below), defects were introduced at locations within the mesh so as to cause as little interference between defect signatures as possible. With this extent of defect separation, all 12 defects could be introduced into the mesh without significantly interfering with any other defects' signatures. Figure 5.16 shows a schematic view of the positions of the 12 defects that were introduced into the mesh, along with their corresponding defect identifications.

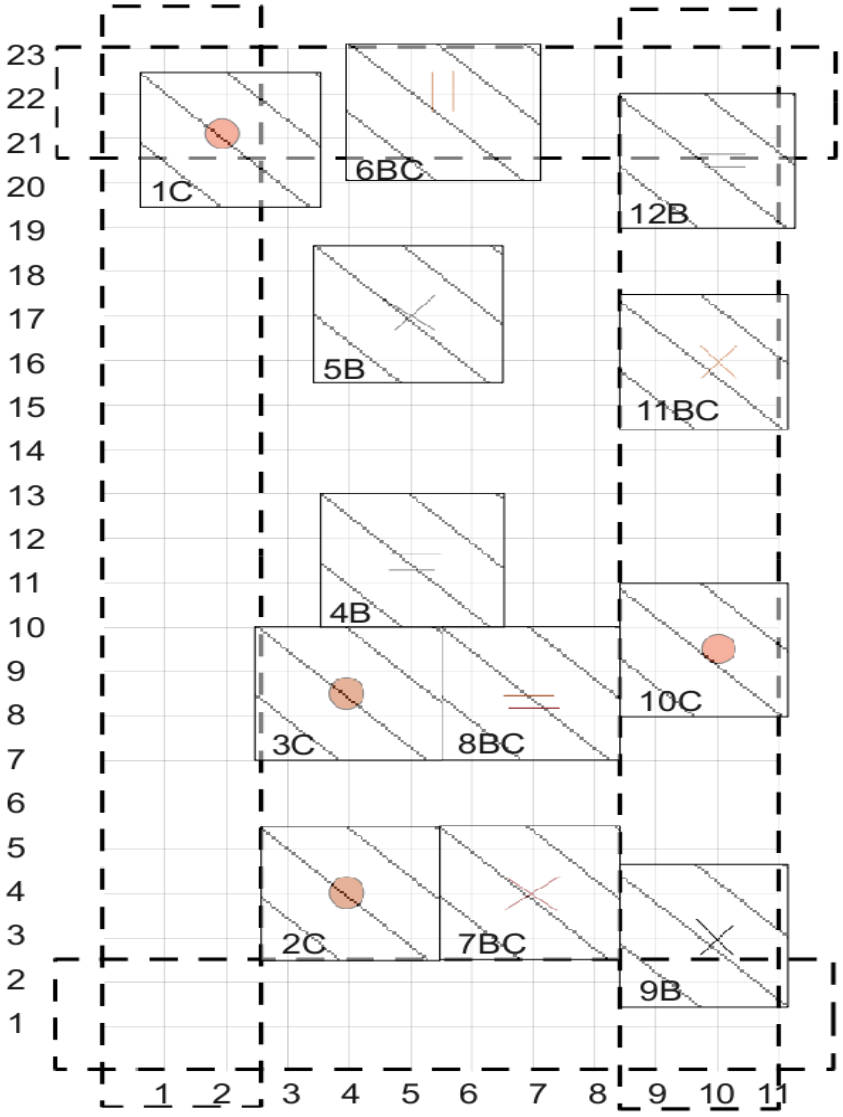


Figure 5.16: A schematic representation of the mesh with all 12 defects and their exclusion zones marked. The dashed areas outline the mesh end effect zones caused by the energisation process. Note that the letters after the defect number denote defects with corrosion (C), breaks (B) and breaks with corrosion present (BC).

5.2.1.5 Introducing accelerated corrosion into the mesh

As outlined in Chapter 1 and shown in greater detail in Appendix A, there are many types of corrosion products which occur as a result of different chemical processes in reinforced concrete. The two main products, which are also magnetic, are magnetite and maghemite. The magnetic properties arising from these two types of corrosion can add to the signal present in EMAD signatures and, therefore, it was decided to try to create these products on the reinforcing mesh in order to replicate corrosion of steel within real-world structures, a process which typically takes many years in the real-world. Once these two products had been added, their progress deeper into the steel was accelerated using a sodium chloride solution. Since this artificial acceleration creates corrosion-based defects faster than in real-world structures, the reinforcing mesh was scanned on a weekly basis to capture the time-course of the developing corrosion. In the real-world, this time-course can range roughly from 15-200 years, but the processes involved in the change from a healthy mesh to a defective corroded mesh are often the same. Therefore, speeding up this process allows for the development of corrosion defects to be monitored in a shorter time period while capturing the time-course of corrosion that is often found in the real-world, without compromising the accuracy of the process of the onset of corrosion within the reinforcing steel.

Introducing corrosion products into the mesh involved heating the target areas for a period of 10 minutes using a heat gun in the range of 250-290°C. This approach was adopted based on the work conducted by Gehring and Hofmeister [331] who found that heating lepidocrocite (a native oxide commonly found on steel) in the range 200-300°C created maghemite; that was then confirmed by infra-red spectroscopy. The authors found that maghemite was formed after heating lepidocrocite in air for one

hour at 250°C. It was also found that a temperature of 273°C created maghemite with the strongest magnetic field, whilst temperatures over 300°C led to a decreased magnetisation in the material. Due to the size of the mesh, it was not feasible to heat areas where corrosion was to be introduced for one hour periods. Instead, selected areas were heated for a period of 10 minutes after which it was shown, through x-ray diffraction (XRD), that the unheated steel collected from the mesh consisted mainly of lepidocrocite, and that after heat was applied the main iron oxides present were maghemite and magnetite, the two desired iron oxides which are often the products of corrosion processes. Further details, including the XRD plots are discussed in Appendix D. Figures 5.17 and 5.18 show two of the areas of the mesh heated to create maghemite and magnetite (note the slightly darker areas of the steel where heat was applied).

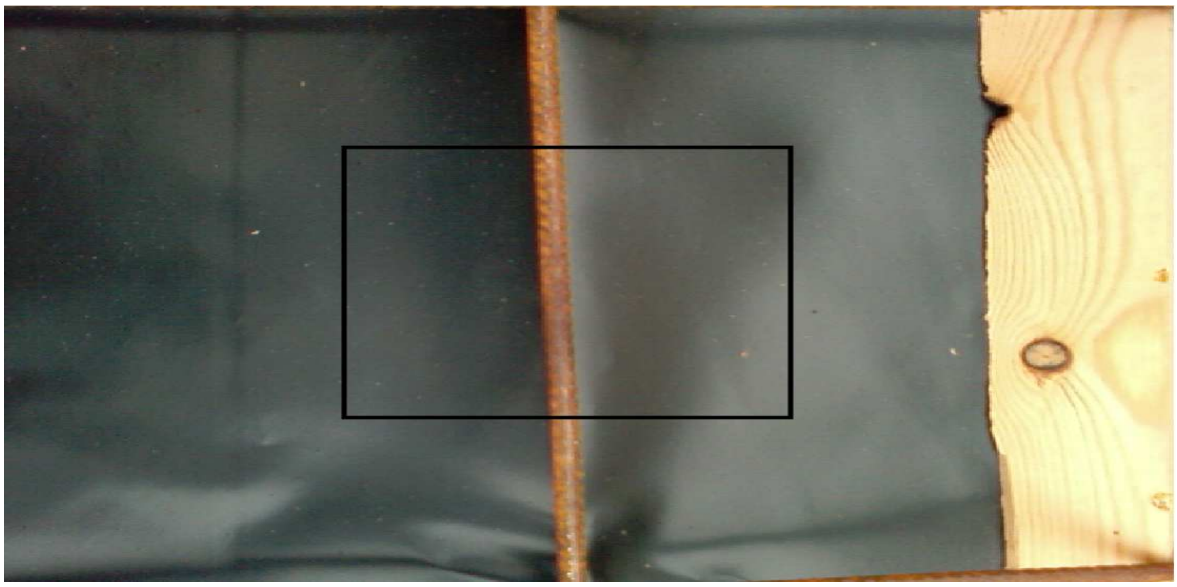


Figure 5.17: An area of mesh heated to create magnetic maghemite and magnetite (centre of the black box).

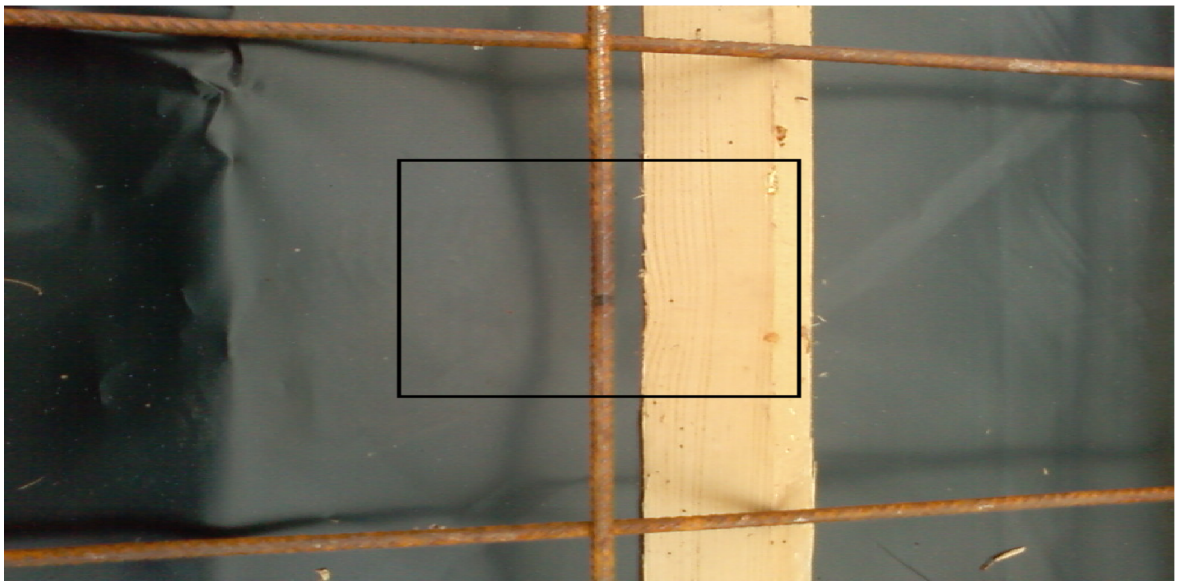


Figure 5.18: An area of mesh heated to create magnetic maghemite and magnetite (centre of the black box).

5.2.1.6 Corrosion tank setup

A corrosion tank was created to form a mould for the concrete and to provide a watertight environment for the soaking of the concrete in the sodium chloride solution. As mentioned previously, wooden ply-boards, which marked the locations of the rebars and trajectories of the scan directions, were placed over the top of the concrete. During mixing, following the procedure by Wheat [332] a high water-to-cement ratio of 0.57:1 was used to increase the porosity of the concrete in order to aid the corrosion process. Figure 5.19 shows the mesh in the tank prior to concrete encasement with some of the marked boards placed over the top of the mesh by way of illustration.

Figure 5.19: The reinforcing mesh inside the corrosion tank prior to concrete encasement with some of the marked boards placed over the top.

Once the concrete was poured, set and had cured under water for four weeks, the concrete containing the mesh was soaked in a sodium chloride solution for a period of two weeks, then allowed to dry for two weeks. This process was repeated throughout the data collection phase over a period of 57 weeks. Following the ASTM G109 standard [333] the water-to-salt ratio was 97 parts water to 3 parts pure freeze dried sodium chloride by weight. This standard also stated that the temperature of the environment of the concrete should be kept within the range of 23 +/- 3°C, but due to the mesh's size, it was housed in a greenhouse where the temperature fluctuated, as shown in Figure 5.20. For a thorough overview of the corrosion tank setup, see Appendix E.

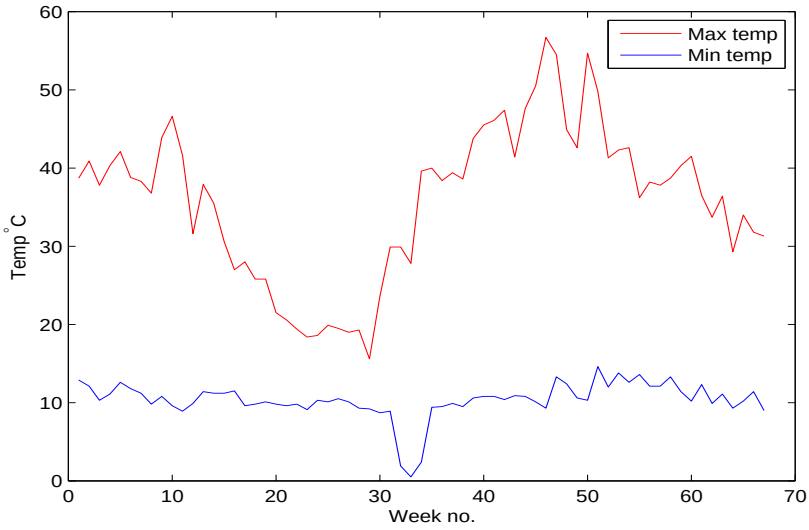


Figure 5.20: The temperature recorded over the duration of the mesh experiment showing the minimum and maximum temperatures which were recorded during weekly data collection.

5.2.1.7 Data collection

As mentioned previously, it was decided to scan the mesh along and between each rebar both in longitudinal and transverse directions, as well as scanning the mesh along paths angled at 30°, 45° and 60°. The mesh was also energised longitudinally or transversely for each of the aforementioned scan directions, giving a total of 14 different datasets for every week of data collection. For each direction of energisation, the polarity of the energiser was the same for every dataset. This section outlines the important features of the dataset prior to and post concrete encasement, along with further investigations into the data collection approaches adopted in order to obtain the best signal-to-noise (SNR) ratio. In this case, the SNR refers to the signal of interest (i.e. the defect) versus the background magnetic properties of the mesh caused by the fringing fields from the structure of the mesh.

5.2.1.8 Data collection pre-encasement

After all of the defects had been created in the mesh, data were collected once prior to concrete encasement, using the EMAD and energisation scanning directions discussed previously. This was to build up a profile of the mesh with defects before it was set in concrete. This would then be compared to data collected after concrete encasement. In order to assess the difference in applying the energiser both longitudinally and transversely, data were collected in order to evaluate which energisation direction gave more insight into each of the defect signatures. Figures 5.21 and 5.22 show data collected from energising and scanning longitudinally and energising transversely and scanning longitudinally along rebar five, while Figures 5.23 and 5.24 show data collected from energising and scanning longitudinally and energising transversely and scanning longitudinally along rebar ten.

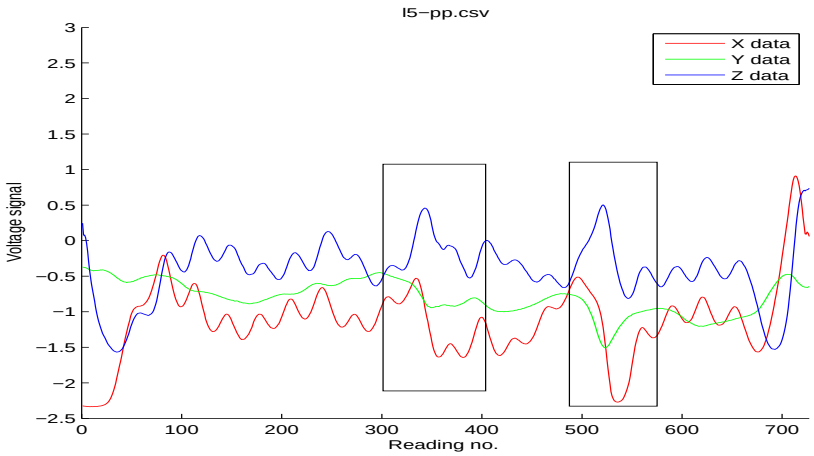


Figure 5.21: Data collected from scanning the reinforcing mesh longitudinally along rebar five after longitudinal energisation of the whole mesh with the signatures of defects 4B and 5B highlighted.

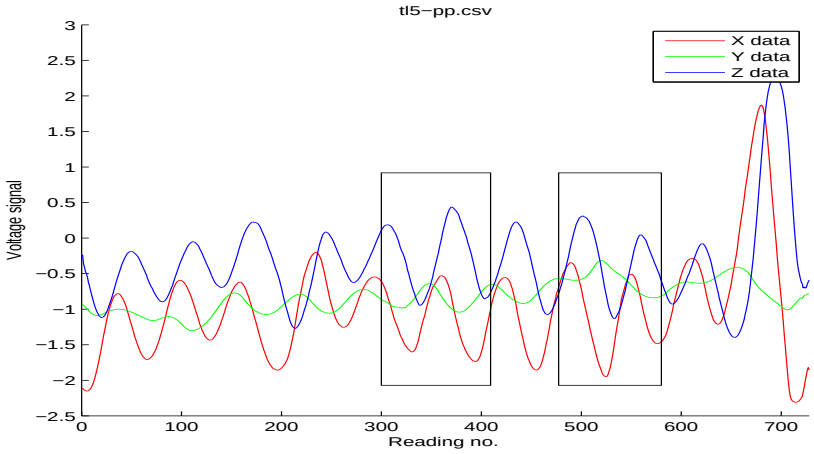


Figure 5.22: Data collected from scanning the reinforcing mesh longitudinally along rebar five after transverse energisation of the whole mesh with the signatures of defects 4B and 5B highlighted.

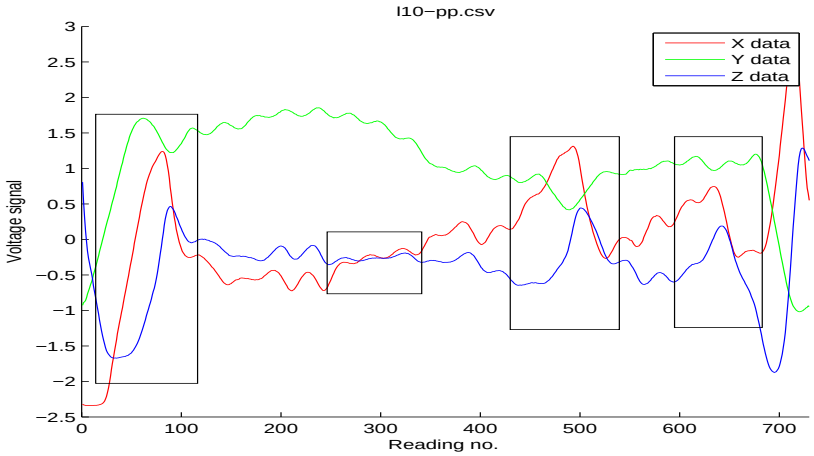


Figure 5.23: Data collected from scanning the reinforcing mesh longitudinally along rebar ten after longitudinal energisation of the whole mesh with the signatures of defects 9B, 10C, 11BC and 12B highlighted.

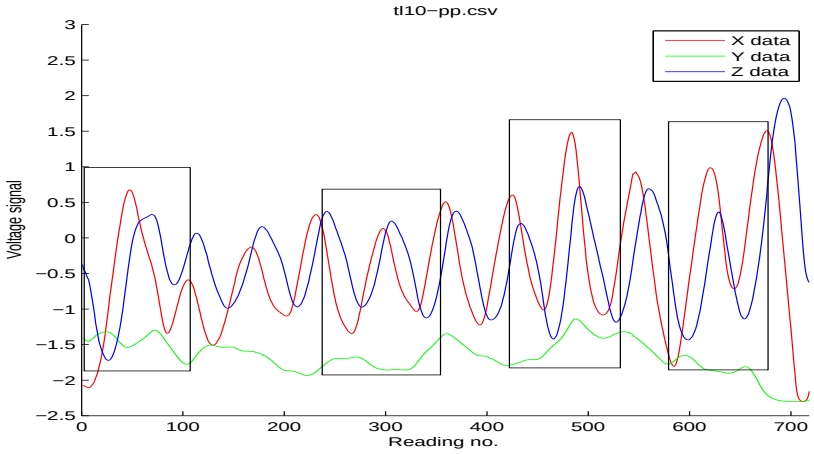


Figure 5.24: Data collected from scanning the reinforcing mesh longitudinally along rebar ten after transverse energisation of the whole mesh with the signatures of defects 9B, 10C, 11BC and 12B highlighted.

Defects can be seen when energisation is applied both longitudinally and transversely, for example see Figure 5.21. Around the 300^{th} and 500^{th} datapoint are the signatures of defects 4B and 5B respectively. However, there are differences between datasets which have been energised in different directions; this is expected as it indicates that the magnetic field applied by the energiser was strong enough to overwrite the magnetisation created during previous energisations, increasing the reliability of the data collected. From Figures 5.21, 5.23, 5.22 and 5.24 one can observe that most defects are clearer when the mesh had been energised longitudinally and scanned in the same direction, as the signals obtained from non-defective rebars are smaller when compared to the transversely energised data. It is for this reason that all of the defects were analysed after energisation in the longitudinal direction and transversely energised scans are not discussed further in this work.

The reason why signals from normal defect-free rebars when energised transversely are larger than when energised longitudinally follows from the influence of the geometry of the structure in relation to the energisation direction. A large ratio between the length of the energisation direction and the other dimensions of the steel (width and height) leads to the creation of smaller fringing fields after energisation. As the mesh is shorter in length transversely, energisation in this direction leads to the creation of larger fringing fields. For this reason, energising longitudinally gives a higher SNR ratio when compared to data which has been collected after energisation transversely.

Figures 5.25 and 5.26 show a plot giving an overview of all 10 longitudinal scan lines after longitudinal energisation before and after all 12 defects were created, with the first scan line at the top of the plot and the tenth scan line at the bottom of the plot. Each scan line in the plots includes a red, green and blue line which represents the electromagnetic properties of the mesh in the X, Y and Z directions respectively.

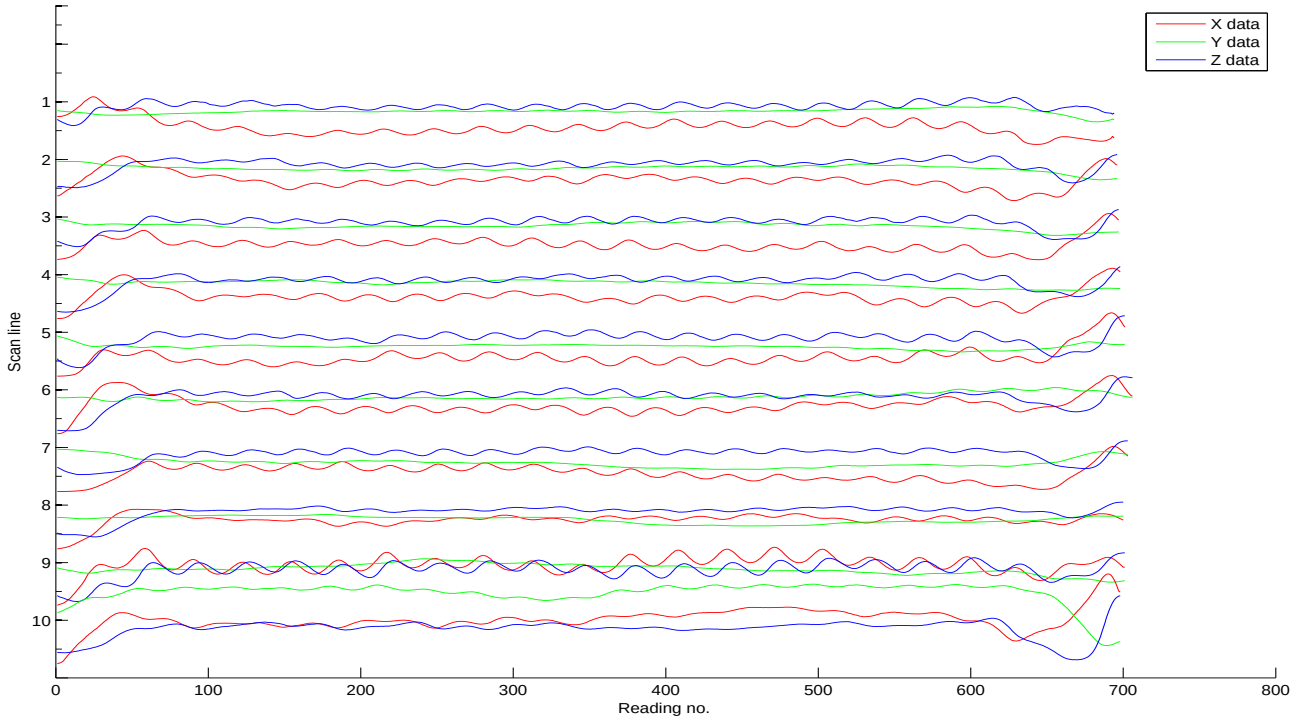


Figure 5.25: A plot of all ten scan lines collected from a healthy mesh prior to defect creation and concrete encasement when energising and scanning longitudinally.

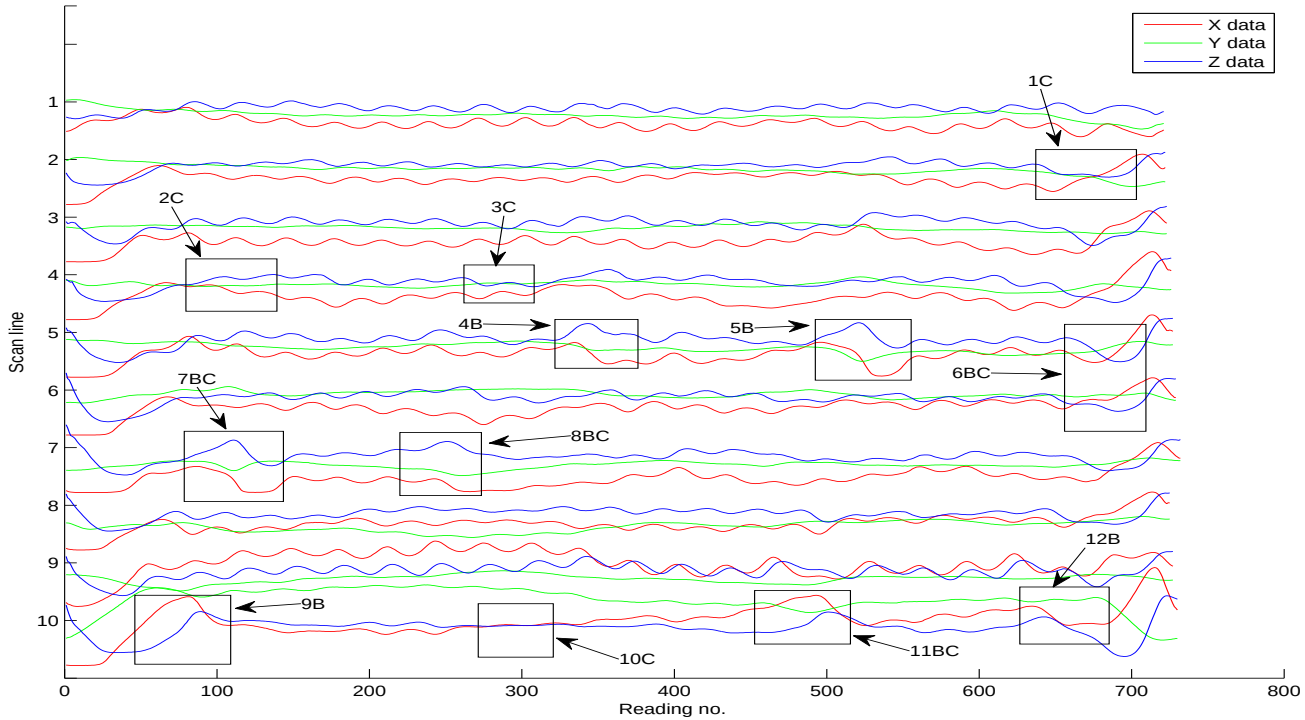


Figure 5.26: A plot of all ten scan lines collected prior to concrete encasement with all 12 defects created when energising and scanning longitudinally as shown by the labelled boxes (note some defects are difficult to detect by eye). In this plot only defects on their corresponding scan lines are labelled for clarity, even though their effects may be visible in neighbouring scan lines.

As Figures 5.25 and 5.26 show, upon first inspection there is a difference in the data collected longitudinally after the defects have been introduced into the mesh. The scan lines in Figure 5.25 follow the characteristic periodic rebar pattern, and this only changes at the beginning and end of scan lines as a result of magnetic dipoles. However, in Figure 5.26 the periodicity of the data changes along certain scan lines, in some cases very drastically, as a result of the introduced defects. An example of this can be seen when looking at scan lines four and five. One can see at around the halfway point on both scan lines a definite change in the data pattern which relates to defect 4B (although it is not labelled on scan line four, it is clearly visible from the figure). In scan line five about two thirds of the way along the scan line there is an even bigger anomaly in the data which corresponds with defect 5B. These plots clearly show the large defects such as breaks, which is expected as a break creates a large fringing field that is easily detected by the EMAD probe. It is, however, rather difficult to see the smaller purely corrosive defects by comparison of Figures 5.25 and 5.26. For figures of each scan line when energising longitudinally and scanning in both directions containing all 12 defects, see Appendix F. Inspection of the data from the end of each scan line reveals that the order in which the rebars are energised has an influence on each rebar's end effect as was discussed previously in section 5.2.1.2. The first few scan lines (see top plots) have smaller end effects than the last scan lines (see bottom plots).

To gain further insight into the EMAD signatures of these smaller purely corrosive defects, individual graphs were studied to try to evaluate the presence of the newly created defects. Three of the longitudinal rebars which included the less apparent defects were rebars two, four and ten as all of these scan lines had examples of corrosion only defects (i.e. defects 1C, 2C, 3C and 10C which contained maghemite and magnetite

artificially created earlier). Further inspection of each of the scan lines containing these defects was conducted (see Appendix G for the detailed analysis) where it was found that defects 1C, 2C, 3C and 10C could be detected when energising and scanning in the longitudinal direction.

Defect 6B, however could not be detected by visual inspection, probably as a result of the presence of end effects at scan lines five and six where the defect was located. Inspection of data collected when energising longitudinally and scanning transversely revealed that defect 6B could be detected by comparing the pre and post-defect data of scan line 22. Therefore, defect 6BC can be seen when energising the mesh longitudinally and scanning it transversely, rather than scanning longitudinally. The reasons for this could be that the magnetic field created towards the end of the mesh obscures the signature of the defect or the fact that the break's orientation was in a different direction from the other breaks, making it difficult to detect when scanning longitudinally. For example, defects 4B and 12B are breaks for which the orientation of the break and the path of the probe when scanning longitudinally are parallel, whereas the break for defect 6BC is perpendicular to the path of the probe. Therefore, it could be argued that in order to detect a break within a mesh-reinforced concrete structure, one should scan the concrete using the EMAD in a direction parallel to the direction of any suspected rebar break. When scanning structures in the real world, where the location and orientation of defects are unknown, it is therefore recommended that, time permitting, the structure should be scanned both longitudinally and transversely in order to detect as many as the defects as possible that might otherwise be hidden either by end effects or due to their orientation.

5.2.1.9 Obtaining the best signal-to-noise ratio for each defect signature

The data shown so far in this chapter have been collected by energising the whole of the mesh in one direction, and then scanning in all scan directions. In order to establish whether this was the best approach to obtain data with the largest SNR possible, a test was conducted in which the longitudinal rebars were energised and then scanned one at a time. This should, in principle, provide a clearer signal from the rebar as energising only a single rebar creates larger fringing fields from that rebar. Energising the whole of the mesh before scanning reduces the larger fringing fields in the previously energised rebars. When energising the whole mesh only the last rebar to be energised has large fringing fields as no subsequent energisation is applied to overwrite its magnetisation. Figures 5.27 and 5.28 show the plots of the whole dataset collected when energising the whole mesh longitudinally and scanning longitudinally and when energising each individual scan line longitudinally and scanning in the same direction before repeating the process on the next rebar.

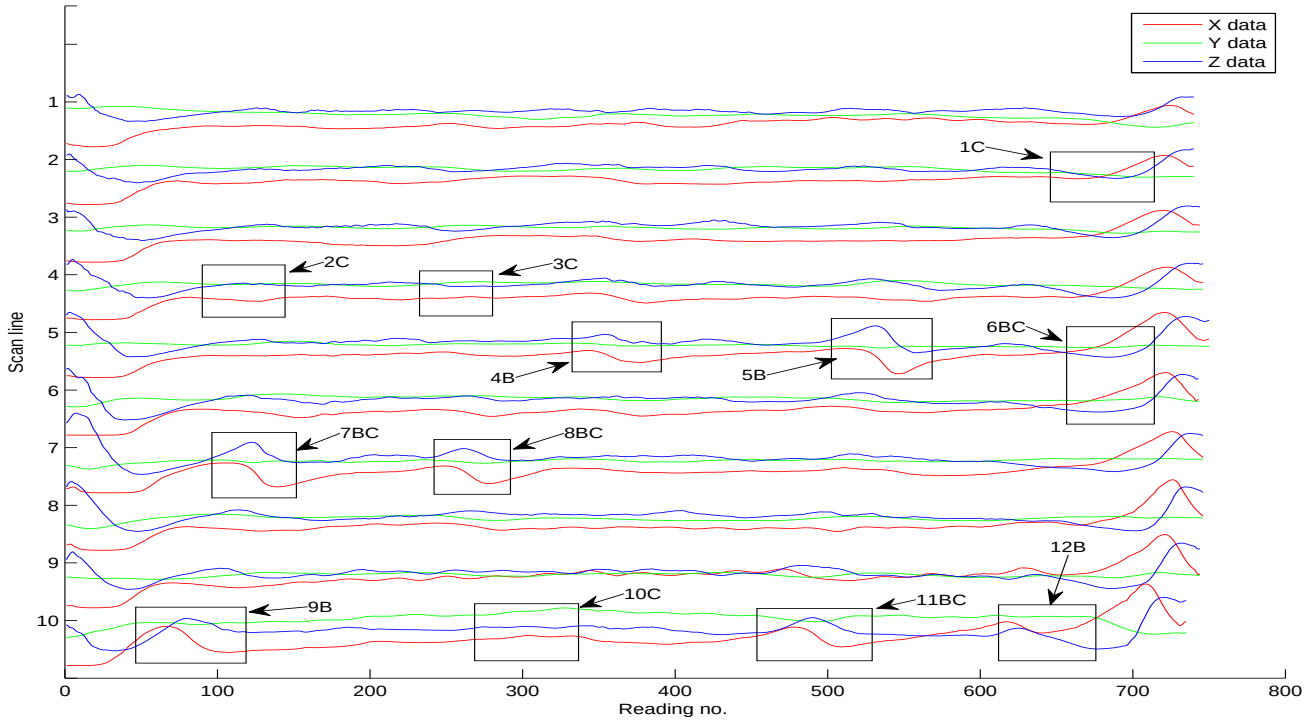


Figure 5.27: A plot of all of the data collected when energising the whole of the mesh longitudinally and then scanning all the rebars in the same direction with the location of each defect shown. For clarity, only defect signatures on their corresponding scan lines are labelled, even though they may be present in neighbouring scan lines.

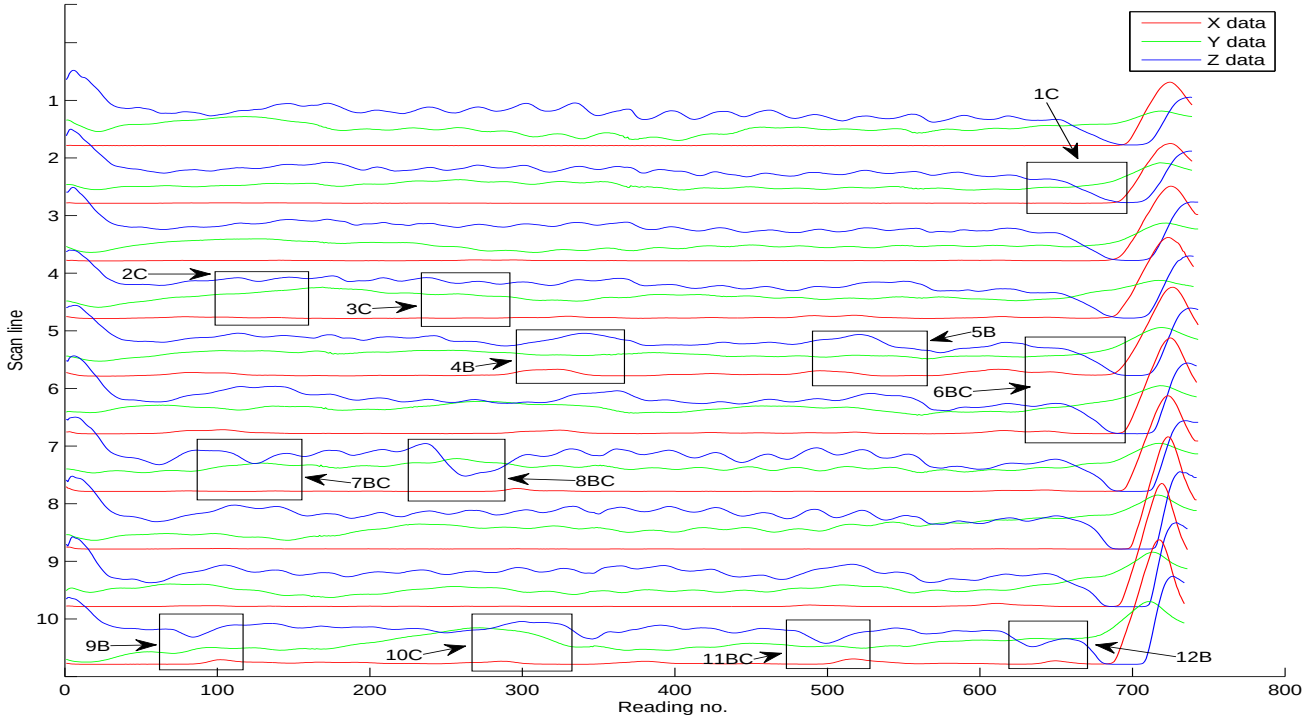


Figure 5.28: A plot of all of the data collected when energising each individual scan line longitudinally and scanning it in the same direction before repeating the process for the next scan with the location of each defect shown. Again, for clarity, only defect signatures on their corresponding scan lines are labelled, even though they may be present in neighbouring scan lines.

Inspection of Figures 5.27 and 5.28, although not as clear as inspection of the individual scans (see below), reveals that energising and scanning each scan line individually actually results in a lower SNR (i.e. the defect signature is less pronounced as a result of increased interference from the background characteristics of the mesh) when compared to energising all of the scan lines and scanning afterwards for the majority of scan lines. A stronger signal is detected from the mesh itself when energising and scanning each scan line individually as was expected, but this has the undesirable effect of increasing the signal from all of the magnetic properties of the mesh, which in term reduces the prominence of the signal from the defect that is of interest. Figures 5.29 and 5.30 show data collected from individually energising and scanning longitudinally and energising the whole mesh and then scanning along only rebar five respectively, which contains two of the most apparent defects, 4B and 5B.

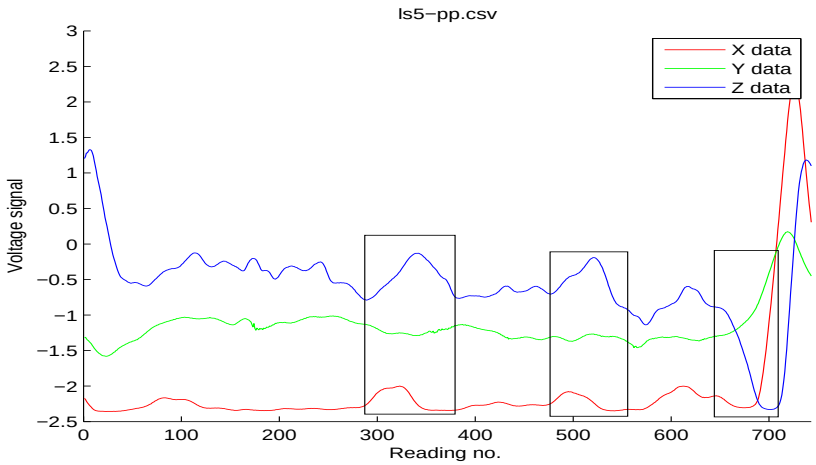


Figure 5.29: Data collected from rebar five when energised and scanned longitudinally individually showing the defect locations of defects 4B, 5B and 6BC.

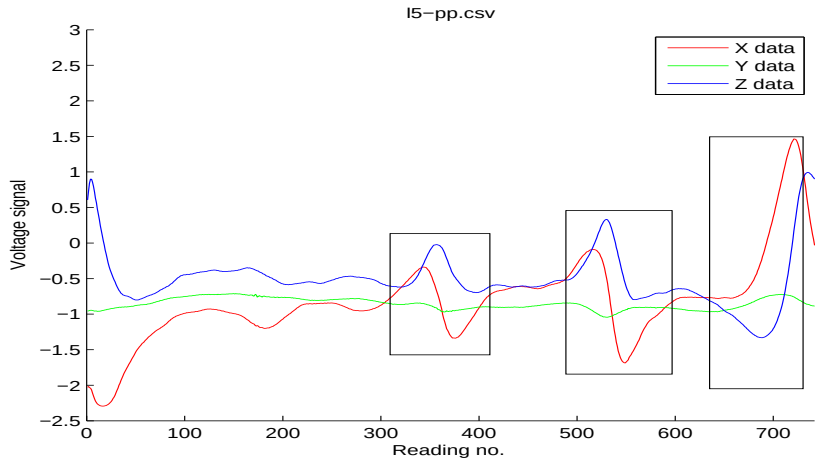


Figure 5.30: Data collected from rebar five when energisation was applied to whole mesh and scanned longitudinally showing the locations of defect 4B, 5B and 6BC.

Comparison of Figures 5.29 and 5.30 shows this effect well, as two of the most prominent defect signatures (which are clearer in Figure 5.30) become less apparent in Figure 5.29 when each scan line has been individually energised and scanned. For all scan lines collected when energising and scanning each individual scan line prior to the next scan line see Appendix H. This finding would probably not have been incorporated into a computer model of the same reinforcing mesh set-up, and hence is an example of one of the benefits of using a controlled laboratory experiment over a computer model. From an engineer's inspection time point of view, these results are also beneficial as it is in fact advantageous that, in the majority of cases, the best signatures can be obtained from a reduced work load since on-site visit time is often restricted (energising and scanning each scan line is more time consuming than energising the whole structure and then scanning the whole structure). This is especially advantageous when using the EMAD as small teams of two to three engineers can conduct a survey of a large area with relative ease.

As the SNR of the EMAD data collected from the last scan line was lower than the other scan lines for the reasons discussed in section 5.2.1.2 and above, applying an energisation field starting from the last rebar (ten) and finishing at the first rebar (one) was also investigated. By applying the energiser in this direction it was hoped that the SNR of scan lines nine and ten would increase, although if the same behaviour observed thus far were to be observed again, the SNR of the last scan line to be energised (scan line one) would decrease. Figure 5.31 shows a plot of the entire dataset collected when energising the whole mesh from the right hand side longitudinally and scanning longitudinally along rebars one to ten as before, while Figures 5.32 and 5.33 show scan line nine when energised using the usual approach from left to right and when energised in the reverse direction from the right side to left side of the mesh. Figures 5.34 and 5.35 show data collected from scan line ten with the usual left to right energisation procedure and the reverse energisation from the right lower corner of the mesh to the left lower corner. For plots of all scan lines collected using the reverse energisation process, see Appendix I.

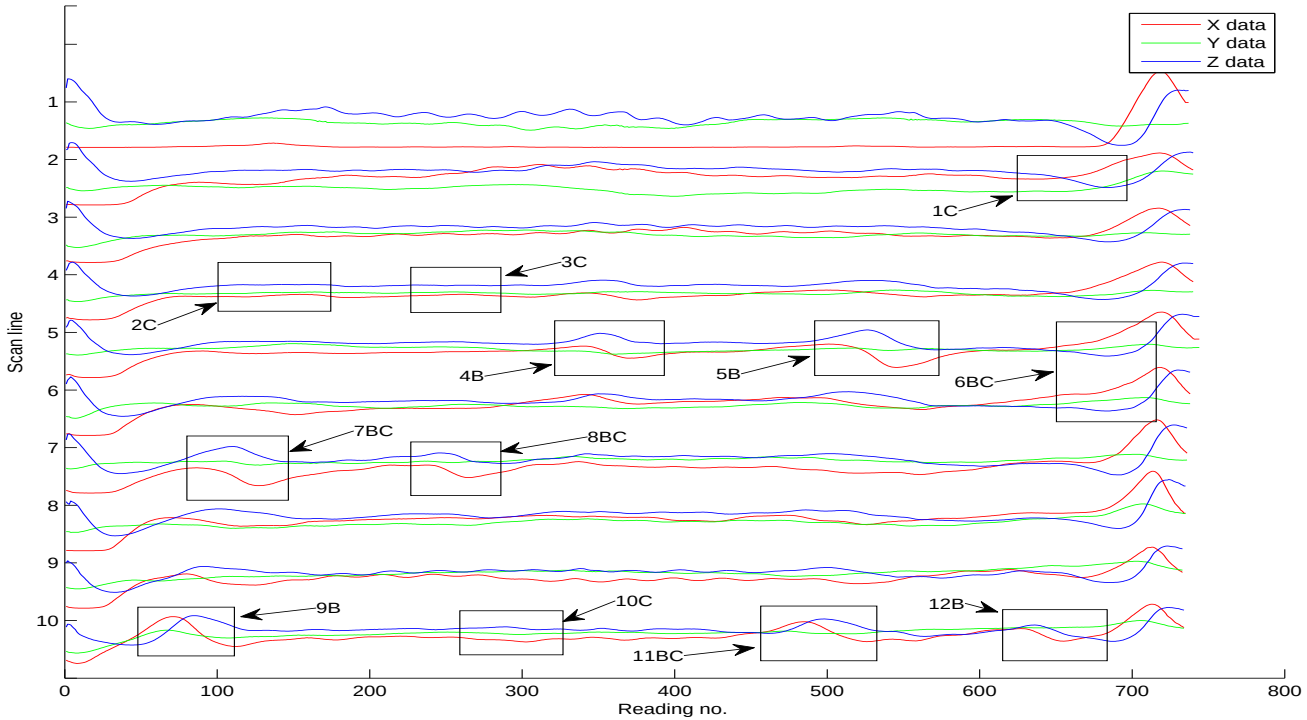


Figure 5.31: A plot of all ten scan lines collected when energising longitudinally from scan ten to scan one with the locations of all defects shown. Note the increased SNR (i.e. visibility of the defect signatures) of the scan lines nine and ten which were obtained as a result of energising these scan lines first. As with previous plots showing whole datasets, only the defect locations for their corresponding scan lines are shown for clarity.

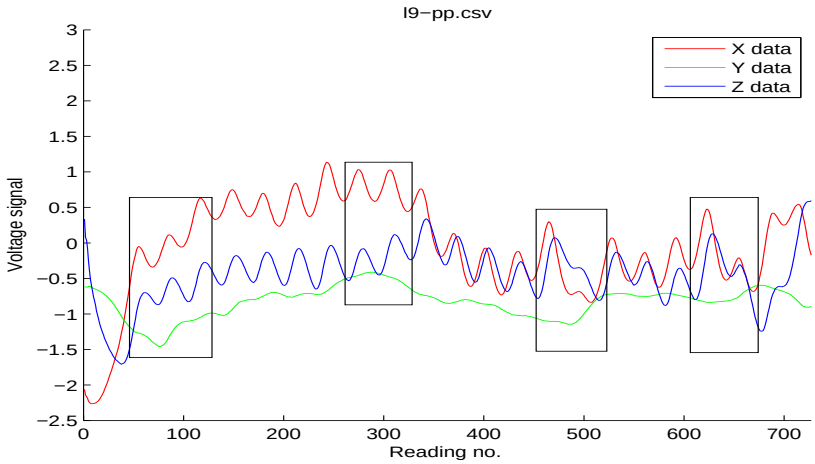


Figure 5.32: Data collected from scan line nine when energising the mesh using the usual procedure starting from scan line one to scan line ten where four introduced defects from scan line ten can be observed.

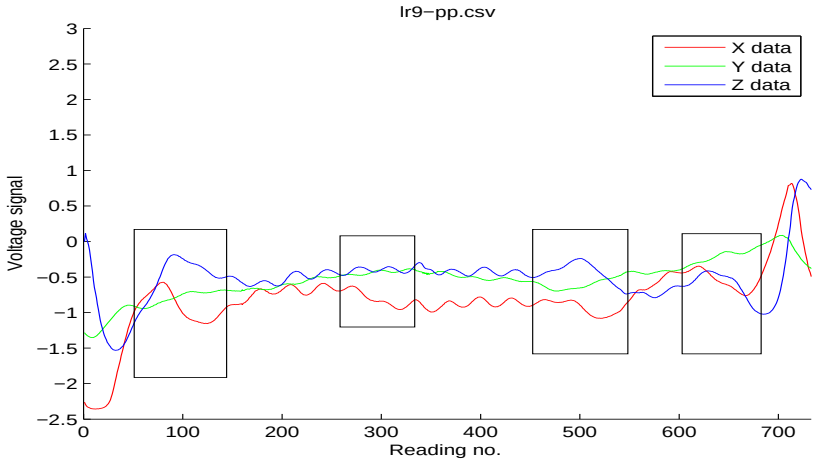


Figure 5.33: Data collected from scan line nine when energising the mesh starting from scan line ten to scan line one showing the four defects present in neighbouring scan line ten.

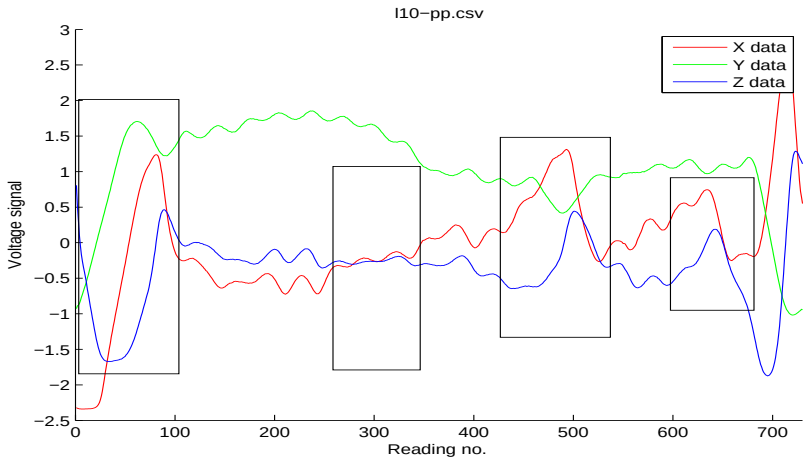


Figure 5.34: Data collected from scan line ten when energising the mesh using the usual procedure starting from scan line one and finishing at scan line ten showing signatures of the four defects present.

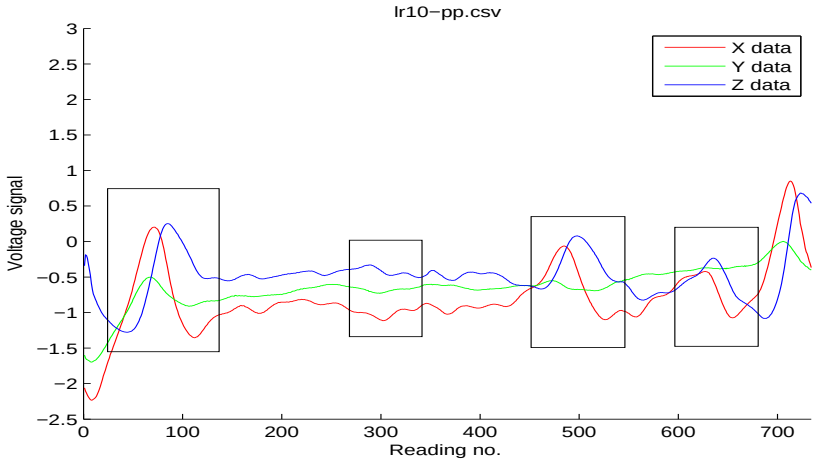


Figure 5.35: Data collected from scan line ten when energising the mesh starting from scan line ten to scan line one showing the same four defect signatures which are much clearer as a result of an increased SNR.

Comparison of Figures 5.32, 5.33, 5.34 and 5.35 shows that energising in the opposite direction (starting from the right side of the mesh and ending at the left side) increased the SNR of scan lines nine and ten, enhancing the defect signatures whilst removing other mesh characteristics and noise. The most prominent improvement can be seen in Figure 5.35 where the signature for defect 10C can be seen much more clearly than in Figure 5.34. Therefore, in order to collect the best signatures for defects 9B, 10C, 11BC and 12BC the mesh was additionally energised starting from scan line ten, finishing at scan line one.

Analysis of the angled scans also revealed that energising and scanning longitudinally gave the clearest signatures for each defect, and hence, the angle scans were no longer analysed and are not further presented in this work. To summarise the best energisation and scan procedure to use, Table 5.3 shows an overview of the energisation and scan directions which give the clearest signatures for all 12 defects.

Table 5.3: The energisation and scan directions which give the highest SNR for each defect signature (r2l denotes energising from the right side to the left side of the mesh, while l2r denotes energising from the left side to right side of the mesh).

Defect No.	**Energisation Direction**	**Scan Direction**
1C	Long(l2r)	Long
2C	Long(l2r)	Long
3C	Long(l2r)	Long
4B	Long(l2r)	Long
5B	Long(l2r)	Long
6BC	Long(l2r)	Trans
7BC	Long(l2r)	Long
8BC	Long(l2r)	Long
9B	Long(r2l)	Long
10C	Long(r2l)	Long
11BC	Long(r2l)	Long
12B	Long(r2l)	Long

5.2.1.10 Data collection post-encasement

To enable fair comparison between pre-encasement and post-encasement EMAD signatures, data was collected in every direction along the same scan trajectory lines as with the pre-encasement scan on a weekly basis. During weeks in which the mesh was soaked in the salt solution, it was drained for half a day to allow data collection and then refilled for another week until the saturation period was finished. Figure 5.36

shows the EMAD data of all 10 scan lines collected from the mesh after a curing period of four weeks.

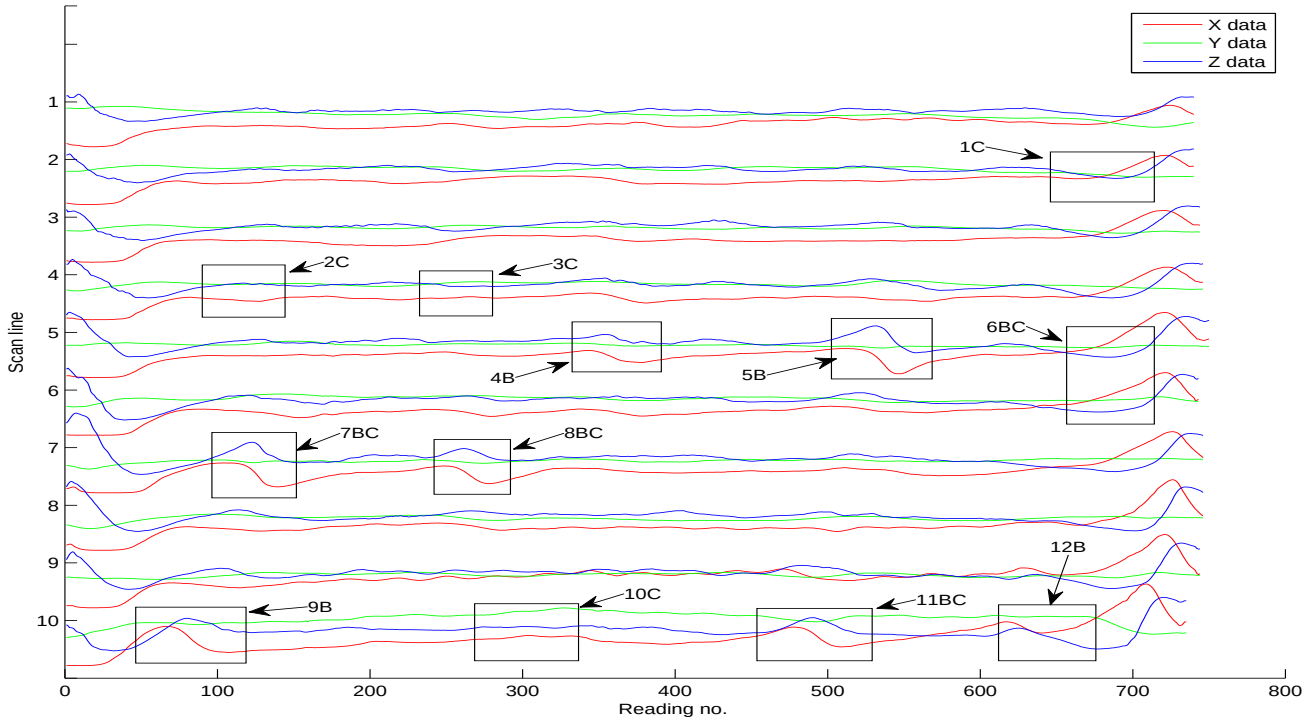


Figure 5.36: A plot of all ten scan lines collected after the steel reinforcing mesh had been encased in concrete with the locations of each defect shown (note the difficulty in visibly detecting some of the defects). For clarity defects are only highlighted on their corresponding scan line even though they may be present in neighbouring scan lines.

Close inspection of Figure 5.36 reveals that signatures of the corrosion-only defects (1C, 2C, 3C and 10C) appear to have disappeared. This could have been due to the formation of a passive layer between the concrete and the steel which may have consumed the corrosion products introduced onto the mesh prior to mesh encasement. Another reason for the apparent disappearance of these defects could be the increased distance between the probe and the mesh as a result of its encasement in concrete. As was shown in the previous section the signatures from these defects were small and required close inspection to become apparent, so increasing the distance between the probe and these defects may have meant that the signatures of these defects were too faint to recognise by eye. Further analysis showed that only defect 10C was observable in the data collected after concrete encasement (a more detailed analysis is given in Appendix K).

The fact that defect 10C can be detected rules out the theory that all of the corrosion products introduced prior to encasement were consumed by the passive layer; therefore the depth of the steel in the concrete (cover depth) may also have had a significant role. This was later confirmed by removal of the concrete during weeks 36 and 39 of the experiment from the areas of the mesh where corrosion was earlier introduced. The concrete was removed from these areas with the aim of reintroducing or exacerbating the areas of corrosion in a shorter time scale than would occur with the concrete surrounding the rebars. Once these areas had corroded sufficiently to be detectable by the EMAD and not consumed by the formation of the passive layer, they would be re-encased in the original concrete mix. Inspection of the cover depth over all corrosion defects revealed that the layer of concrete above the mesh towards the right of the slab was slightly lower than the concrete cover towards the left of the slab. This may explain why defect 10C can still be detected in the data post-encasement, whereas defects 1C,

2C and 3C which have a greater concrete cover cannot be detected in the EMAD data. While this was an undesirable characteristic of the concrete slab design, it is actually more representative of real-world concrete slabs where the cover depth is often found to be inconsistent.

As the areas of the mesh where corrosion was introduced were now exposed to the environment again, it was decided to further exacerbate their corrosive onset by applying hydrochloric acid (HCl) to these areas during the cyclic two week periods of drying. It was hoped this procedure would reduce the time taken for the corrosion signatures to either reappear, in the case of defects 1C, 2C and 3C, or increase their signal strength, in the case of defect 10C.

5.2.1.11 Defects used for training and testing neural networks

Figures 5.37 and 5.38 show all ten scan lines from the first week and final week of data collection respectively with the visible defects labelled for comparison. Here defects are labelled wherever they are visible, even when in neighbouring scan lines. It is the labelled defects in Figures 5.37 and 5.38 that were used as training and testing data for the ANNs whose performance is shown in Chapter 6. For a full comparison of data collected when energising and scanning longitudinally from the first week and final week of the experiment, see Appendix L.

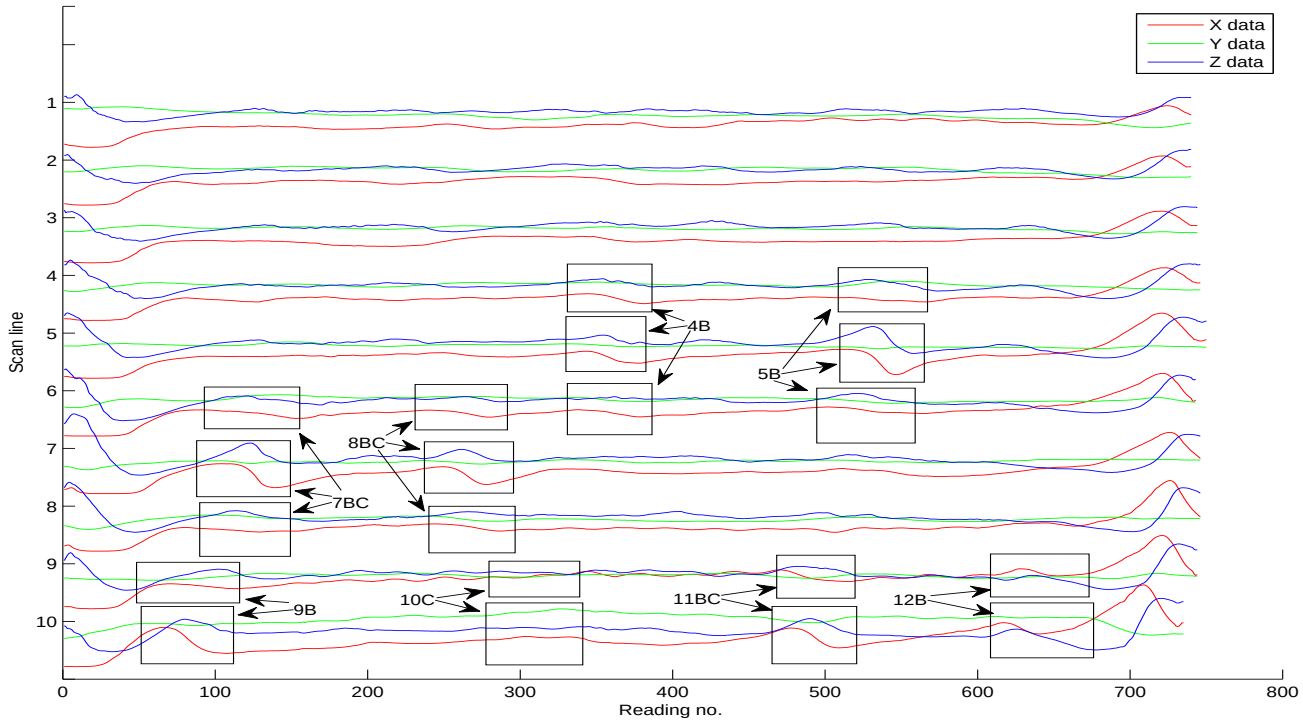


Figure 5.37: A plot of all ten scan lines collected from the first week of data collection after the steel reinforcing mesh had been encased in concrete with the locations of each visible defect shown. These defect labels were used as target defect locations for the training and test data presented to the ANNs for automated analysis.

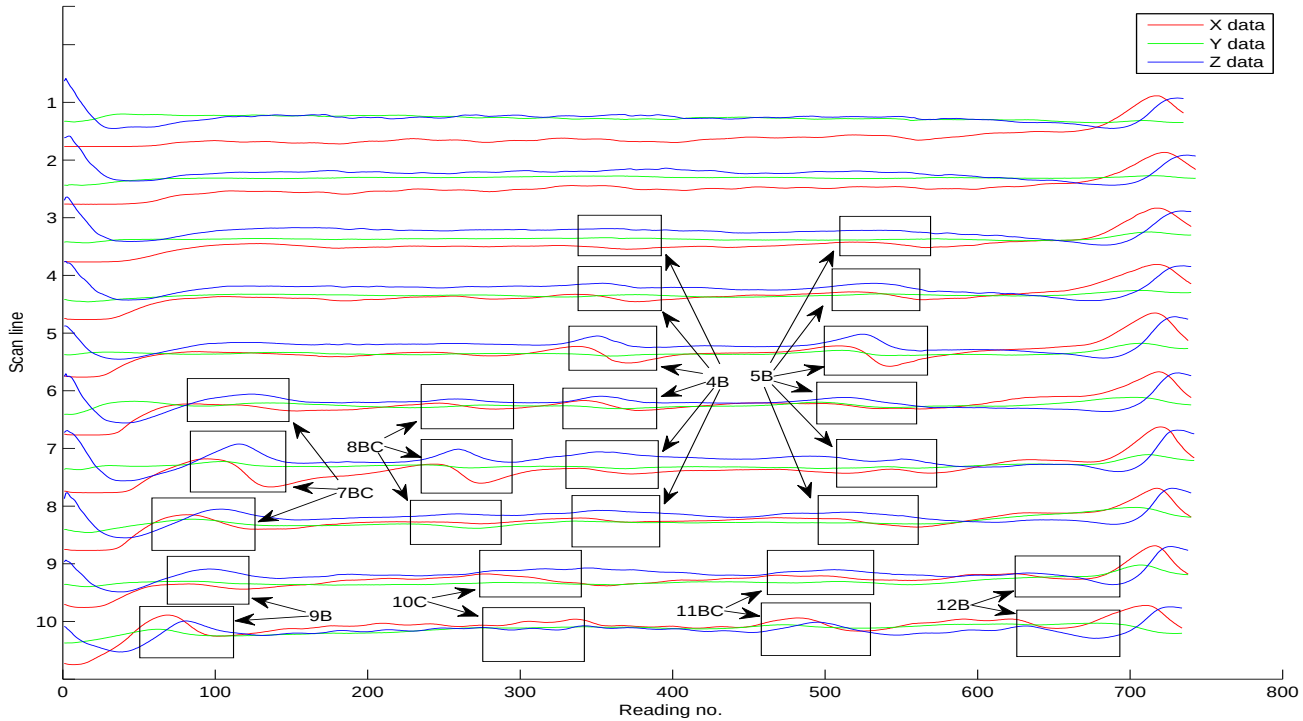


Figure 5.38: A plot of all ten scan lines collected from the last week of data collection after the steel reinforcing mesh had been encased in concrete with the locations of each defect shown. These defect labels were used as target defect locations for the training and testing data presented to the ANNs for automated analysis.

Comparison of Figures 5.37 and 5.38 reveals that unfortunately the corrosion defects which disappeared after concrete encasement had not reappeared in the data at the end of data collection despite the removal of the concrete and the application of hydrochloric acid. For this reason the planned removal of the concrete from the mesh was not conducted as the ground truth defects are still known as a result of little change of the defect signatures over the experiment duration. With the concrete still intact, the experiment will carry on after this research project until sufficient signals from these areas can be detected. The main reason for the lack of change in the defects is probably the short time-course over which the experiment has been conducted. Although the reinforcing mesh was encased in concrete for 67 weeks, four weeks were spent curing the concrete, giving a total of 63 weeks for corrosion exacerbation. Had the corrosion products introduced prior to concrete encasement not been mostly subsumed by the formation of the passive layer or the increased distance between the rebars and the EMAD as a result of the concrete cover depth (with the exception of defect 10C whose signature became slightly clearer towards the end of data collection), these defects may have become gradually worse and pronounced with an increased SNR during the experiment. However, these areas of corrosion were broken out from the concrete during week 36 (for corrosion-only defects) and week 39 (for the remaining defects containing corrosion) of data collection and hydrochloric acid was applied to each one once a week from week 47 for the remaining 20 weeks. While one may expect some corrosion to occur during this period, it is clear from the data that this was not a sufficient amount of time for detectable amounts of corrosion to take place. With more time, it is expected that the defects will become much worse and more pronounced in the dataset.

Analysis of the two figures also shows that while the majority of the corrosion defects are not apparent to the naked eye in the datasets, the defects which were detectable after concrete encasement are still visible. In the case of defects 4B and 5B, one can see that their signature is now much stronger, and as a result visible from scan lines further away, than in the first week of data collection. The reasons for this could be that these areas of the mesh have deteriorated during the experiment creating a larger signal, or that the fringing fields have become stronger during the experiment as a result of successive energisations. While this is an undesirable effect, without a de-energisation process which was not available during this experiment, it was unavoidable. One can also see that defect 6BC is still subsumed in the end effect signal of scan lines five and six and hence is not labelled in the longitudinally collected data.

Closer inspection of scan line four from the first and last week of data collection reveals a small signature of a defect that may relate to defect 3C as shown in Figures 5.39 and 5.40. Since this is a possible indication that this corrosive defect may be detectable, the fact that this is a small signal and was only visible in the data collected from the last few weeks means that it is too early to determine whether this is in fact this defect or just noise from the probe. As the mesh experiment will be ongoing beyond the completion of the current work, it will be interesting to see if this signal becomes larger.

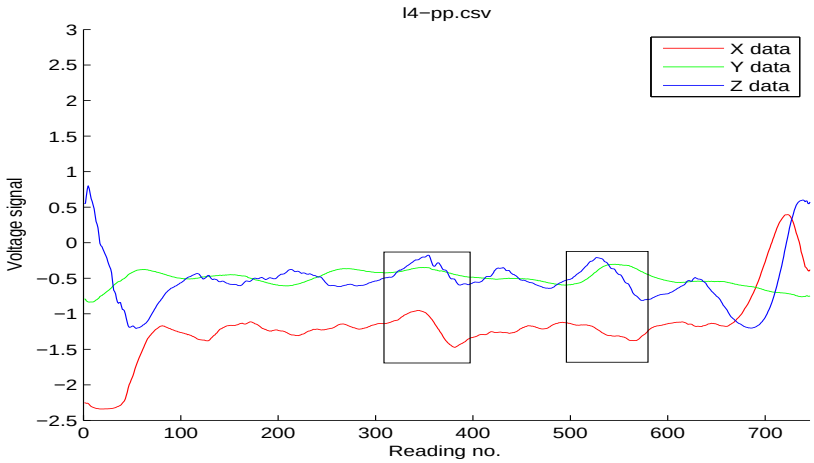


Figure 5.39: Data collected from the first week of data collection showing scan line four when energising and scanning longitudinally. Defects 4B and 5B from neighbouring scan line five are visible and hence are labelled.

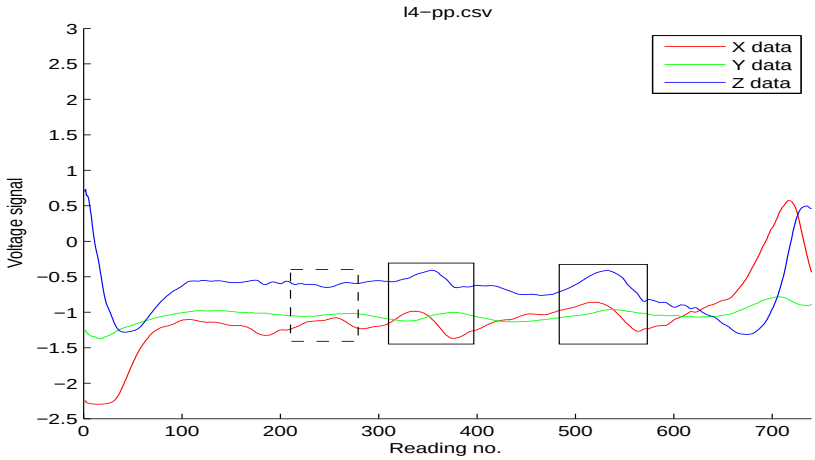


Figure 5.40: Data collected from the last week of data collection showing scan line four when energising and scanning longitudinally. Defects 4B and 5B from neighbouring scan line five are still visible and hence are labelled. A possible signal from defect 3C could be visible as highlighted by the dotted box, although more data is needed to verify this.

5.2.1.12 Defect signatures over the duration of the experiment

As the previous section showed, there was little change observed when comparing the data collected during the first week of data collection to the data collected during the last week, with the major change coming from the increase in the spatial extent over which defects 4B and 5B could be detected. For further analysis of each defect signature throughout the duration of the experiment its signature at five week intervals was analysed. This section shows data collected from scan line five where defects 4B and 5B were located when energising and scanning longitudinally over five week periods which is shown in Figure 5.41 below. For an overview of each scan line over the duration of the experiment see Appendix M. The weeks of data collection indicated on the vertical axis of Figure 5.41 show intervals of five weeks, with the exception of after week 31 where the next week shown is week 38. Readings resume at week 38 as no data was collected in weeks 36 and 37 due to a technical fault with the EMAD hardware.

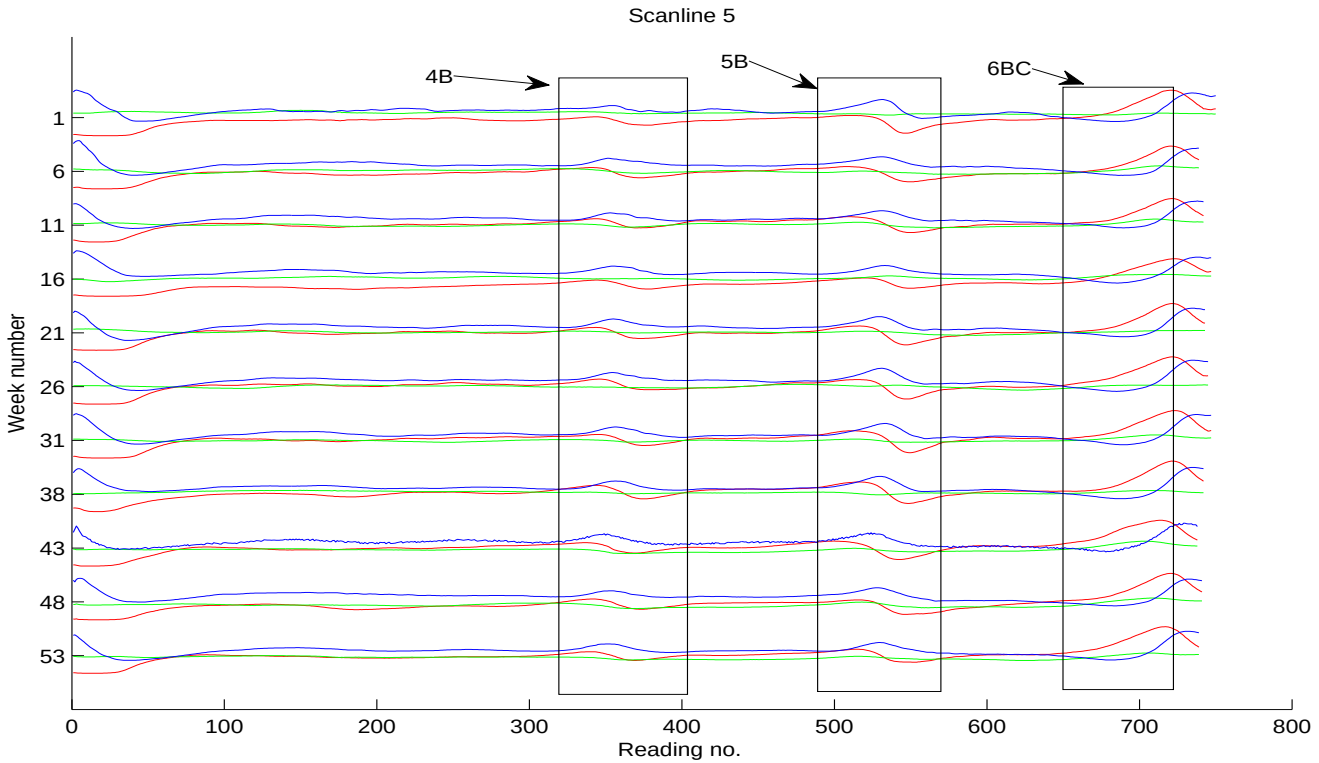


Figure 5.41: Data collected from scan line five over the duration of the laboratory controlled experiment which contained defects 4B and 5B as highlighted. As in the other EMAD plots, the X, Y and Z data are the red, green and blue plots respectively and are between the range of -2.5 to 3 Volts.

As Figure 5.41 shows, the signatures of defects 4B and 5B collected from scan line five experience little change over the course of the experiment. The figure also shows that the signature of defect 6BC is not visible throughout the experiment when energising and scanning longitudinally as it is overwritten by the end effect of the mesh located at the end of the scan line. As expected by the little change in the defects that were introduced onto the mesh prior to concrete encasement, the data collected from scan line five does not show any new defects towards the end of the experiment when compared to the data collected at the start of the experiment.

5.2.1.13 Summary

This section outlined the design of a laboratory controlled experiment to create typical defects found within reinforced concrete slabs and capture their development over time using the EMAD approach. A total of 12 defects were created, each with different physical and spatial characteristics. Although all the introduced defects were detectable prior to concrete encasement, three of the smaller corrosion defects were difficult to detect from the data collected post-encasement. While this was undesirable, with the application of hydrochloric acid in addition to the sodium chloride solution it was hoped that these defects would reappear later on in the experiment. As one can see by comparing Figures 5.37 and 5.38, this did not occur in the time scale of the experiment. The reason for this, despite the exposure to a corrosive environment using the salt solution and acid, is that the corrosion that was being created in this experiment typically takes many years to develop in real-world structures and using the aforementioned approaches to speed up the onset of corrosion the rate was not high enough for the time scale of the research project. Several experiments were also outlined which helped to determine the energisation and scanning procedures which

gave the best SNR for each defect signature. This data was extremely useful as it provided ground truth signatures of defects which are often not available in real-world structures where the type of defect and its extent is difficult to assess.

In total, eight of the 12 original defects were detectable when energising and scanning longitudinally, these were 4B, 5B, 7BC, 8BC, 9B, 10C, 11BC and 12B. As one can see, all of the break defects, with the exception of 6BC which is subsumed by the end effect signals of scan lines 5 and 6, can be detected. Only one of the corrosion only defects can be detected, which may be due to the small concrete cover over this defect when compared to the other corrosion only defects. This ground truth data was then used to train different ANNs to automatically recognise these different defects, the results of which are outlined in Chapter 6.

5.2.2 Temple Moor High School

This data was collected from an ageing school building, where many other buildings on the same site had been replaced by modern alternatives, but this block was planned, if structurally sound, to be refurbished and re-opened. The client wanted to obtain insight into the condition of the first and second floors of the building, which were unusually constructed using clay pot encasement of steel tendons which are often used in the construction industry in mainland Europe. Figure 5.42 shows a section of the underside of a floor. The narrow clay pots each contained four to five steel reinforcing tendons encased in grout which ran the width of the floor and were tightened and secured in the outer walls of the building. The wider clay pots adjacent to the narrow pots were empty. As the picture in Figure 5.42 shows, a row of wider pots was followed by a row of narrow pots containing the post-tensioned tendons. This pattern repeated throughout most of the structure, although there were some small areas which con-

tained either all narrow or wide pots, but the pattern soon returned to an arrangement of alternating widths. Figure 5.43 shows a view of one of the floors scanned using the EMAD technique.

Figure 5.42: A section of the underside of a floor of the assessed building. Notice the broken narrow pots with missing tendons as highlighted by the black boxes.

Figure 5.43: One of the floors scanned at Temple Moor.

It was apparent upon visual inspection that the floors had been severely damaged over the building's lifetime. In many places, installation of service pipes had led to cuts through sections of tendon, removing any support they once offered. Figure 5.44 shows an area typical of the condition of a section of a floor which had been damaged due to poor installation or maintenance works. For some time over its life, the upper floor had been used as a science laboratory, which probably explained the many signs of previous water leaks. In addition to this, since the beginning of the school refurbishment parts of the roof had been removed which allowed rainwater to run through sections of both floors; indeed, at the time of inspection several areas of the floors were saturated. Although many defects were apparent upon visual inspection, it was not clear how the damage caused by poor installation and water exposure had affected the parts of the floors which appeared to be in good condition from the surface.

Figure 5.44: An area of the floor damaged by the insertion of a service duct (right) and two pipes (centre).

The first and second floors of the building were scanned from the underside (i.e. the ceiling of the floor below) as the tendons were closer to the surface of the pots on the ceiling side than the floor side. This would ensure clearer, stronger signals during data collection. First, the energiser was applied to the pots containing the steel tendons to put the magnetic field from the steel into an interpretable state. The row of pots containing the tendons was then scanned. Figure 5.45 shows an example of the data collected from the second floor, while Figure 5.46 shows the corresponding section of the row of pots scanned. Upon inspection of Figure 5.45 it is apparent that the data changes very quickly from one time step to the next, with some large peaks present. This is often a sign that an anomaly exists in the structure. Around the 700^{th} datapoint the signal becomes very large in the X and Z directions, indicating a problem with the tendons in the structure. The highlighted area of Figure 5.46 shows the area to which this data point corresponds, showing broken tendons, thereby confirming the

area in the data as a defect. Note the smaller peak around the 800^{th} datapoint which corresponds to the break in the tendons shown bottom centre. For comparison, a scan line with relatively healthy tendons is shown in Figure 5.47 where one can observe less variations and peaks in the data. The data collected was labelled after expert analysis. This labelled data could then be used to train and test different ANNs, the results of which are shown in Chapter 6.

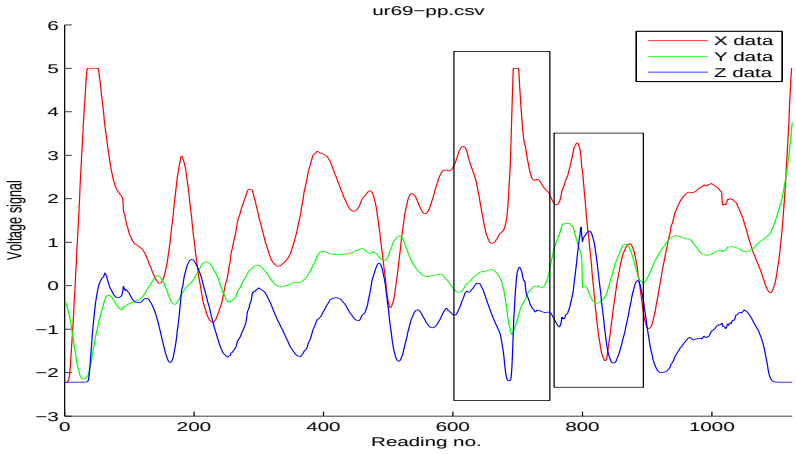


Figure 5.45: An example of the data collected from the second floor highlighting the two sections of data which correspond to the areas where damage was visible from the surface as shown in Figure 5.46 below. As shown by the constant changing of the EMAD signal, other defects may be present around a third of the way along the scan line, although given the magnitude of the signals received from the visible defects, these defects are likely to be less severe.

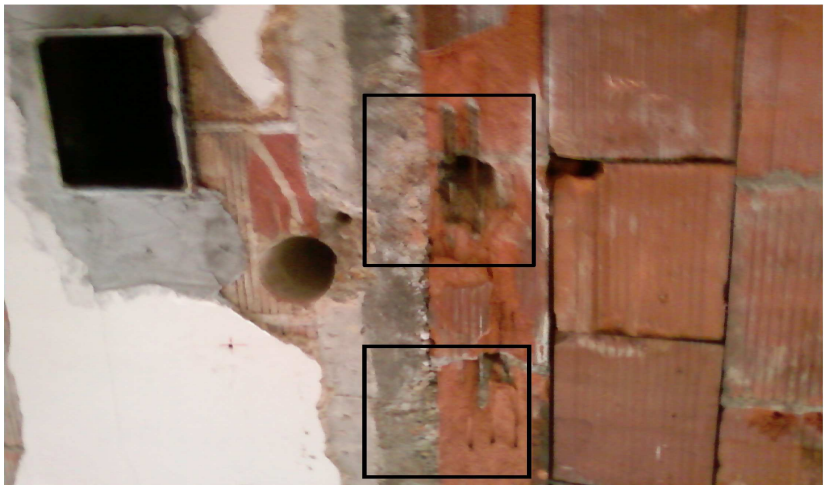


Figure 5.46: A close-up of the corresponding two areas represented in the data around the 700^{th} and $800th$ datapoints shown in Figure 5.45.

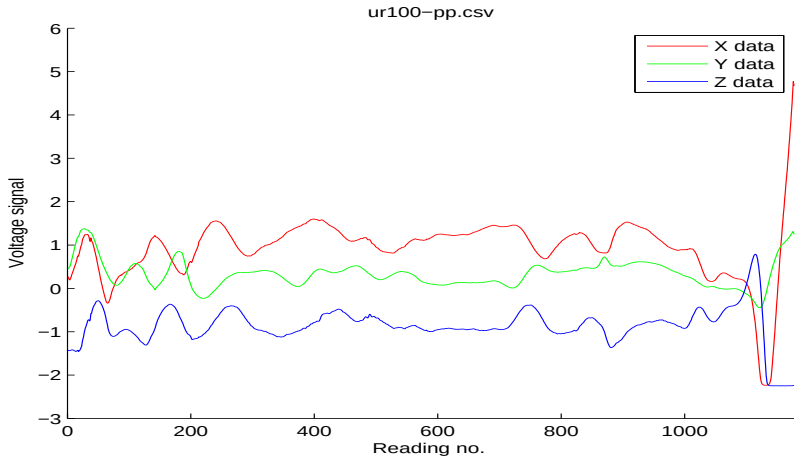


Figure 5.47: An example of the data collected from a relatively healthy part of the second floor of the Temple Moor building which in comparison to the data shown above in Figure 5.45 contains less peaks and variations.

5.3 Summary

This chapter has introduced several case study datasets which will allow comparison of an ESN, the novel R^2SP and the TD-ELM. The first four case studies presented in this chapter come from domains outside of defect detection within reinforced concrete, with the first three datasets created from mathematical models. The fourth order polynomial dataset is a non-linear dataset where a network is required to reproduce an output according to a polynomial, given random input data. This polynomial output also requires memory of the previous input, therefore requiring some very short-term memory of the input data from an architecture. The extended polynomial dataset is a novel extension to the fourth order polynomial which allows the levels of non-linearity (defined by the order of the polynomial) and short-term memory to be changed according to the users requirements. This enables a more in depth analysis of the characteristics of a network, with the task of reproducing the polynomial becoming more difficult as the levels of non-linearity and short-term memory requirements simultaneously increase.

The last of the datasets created from mathematical models is the extended delayed XOR dataset which, as with the extended polynomial, allows the non-linearity and short-term memory requirements of the task to be varied. This dataset differs from the extended polynomial as a network is required to give an output of the binary XOR of three binary bits, rather than a polynomial. This task has been investigated previously in the literature, where it was shown to become very difficult after relatively small increases in the non-linear mapping and short-term memory requirements.

The spoken digit task was the first of the real-world datasets presented in this chapter which consists of 10 spoken utterances of 10 digits, each of which was spoken by five

female speakers. This dataset has been widely studied in the literature, therefore providing multiple comparative results. To make this classification task more difficult, noise taken from a cafeteria was added to each utterance with various levels of SNR.

The final two datasets presented are real-world data collected using the EMAD probe. The first was collected from a steel reinforced concrete slab that was created under laboratory conditions, while the second was collected from a real-world structure whose condition was unknown. These datasets are non-linear (or highly non-linear in the case of the second concrete dataset) and require some short-term memory of previous datapoints. The results of applying an ESN, R^2SP and TD-ELM, as well as RANNs trained using conventional algorithms and state-of-the-art architectures are presented in the next chapter.

Chapter 6

Results

6.1 Introduction

This chapter presents the results obtained when applying a standard RC architecture, the R^2SP and TD-ELM approaches, as well as several other benchmark techniques to the case studies which were described previously.

6.2 Experimental set up

To train of all the reservoir and R^2SP networks used in this research, 10-fold cross-validation using a grid-search was used to find the optimal regularisation parameter for ridge regression unless otherwise stated (both regularisation and ridge regression were introduced earlier in Chapter 3). Cross-validation removed the danger of training and testing using unrepresentative portions of the dataset, while the grid-search ensured that the network did not over-fit the dataset, which often reduces performance on unseen test data. While this search increases the training time of a network architecture, the improved performance gained from a regularised network made the increase

worthwhile.

6.3 Comparative results

In order to find the best parameters of an ESN and an R^2SP when applied to each case study dataset, a broad parameter sweep was conducted using the RCToolbox, outlined earlier in Chapter 4, ranging over: the input scale, spectral radius, leak rate, bias to reservoir neurons and reservoir size. To reduce the search space of the parameter sweep, the number of neurons used was set at a task dependent amount which was determined after some preliminary searching. This involved increasing the number of neurons in the reservoir by 50 at a time. Once performance appeared to stop increasing, the number of neurons was set at this number of neurons. In the case of the R^2SP the number of neurons in the two ELM layers were made equal to the number of reservoir neurons to reduce the search space for parameters. While this may not be the best approach to adopt, the size of these layers would later be tuned using the pruning approach. The remaining parameters were then ranged over to find the values which gave best performance. The number of neurons which allowed optimal performance for each architecture was found later, using the pruning algorithm outlined in section 4.4.1, the results of which are given in section 6.4.

In order to overcome the variations in performance due to the random initialisation of the ESN and R^2SP networks, 100 networks with the parameter settings that enabled best performance were simulated for each dataset. In the results reported below, the errors were averaged and their standard deviations are shown in parentheses. As with the ESN and R^2SP architectures, 100 ELMs were simulated using the optimal window size to reduce any variation obtained through random weight initialisation,

with the test error given as the average with the standard deviation shown in parentheses. In the case of the more conventional RANNs, as a result of their slow and complex training procedure only five networks were created and trained with their average errors and standard deviations also shown. While this is admittedly a point of weakness for comparison, the advantage of reservoir and ELM training when compared to the conventional RANN training methods becomes clear. An RANN which offers at least comparable performance to a reservoir or ELM based approach can potentially be obtained, but their slow and complex training procedures make this difficult. The following sections outline the performance of these architectures when applied to different datasets, along with the values for their tuned parameters and, in the cases where the ESN and R^{2}SP architectures were used, further analysis into their properties has been performed.

6.3.1 Fourth order polynomial

After conducting a sweep over the various ESN and R^{2}SP parameters it was found that varying the size of the ESN and the R^{2}SP-reservoir, $stat_1$ and $stat_2$ layers of the R^{2}SP above 200 neurons did not have a significant impact on their performance so a size of 200 was chosen for each layer. Different values of the leak rate were found to make little difference for this task, which was probably due to the current input depending only on the immediately preceding input; therefore the additional memory provided by changing the leak rate was not required.

Training of an Elman Network (EN), Layer Recurrent Network (LRN), Distributed Time-Delay Neural Network (DTDNN) and a Time-Delayed Extreme Learning Machine (TD-ELM) was also conducted for this task. As the training algorithm for the EN, LRN and DTDNN neural networks had an adaptive learning rate, this was opti-

mised during training. A momentum term of 0.9 was also used with training lasting for 300 epochs. In the case of the Back-Propagation Through Time trained EN (EN-BPTT), a learning rate of unity and a momentum term of 0.09 was used during training, while a learning rate and momentum term of 0.5 and 0.09 were used for the Real Time Recurrent Learning trained EN (EN-RTRL). The best number of hidden layers and neurons in each layer for these networks were found using a trial and error based search where different numbers of layers and layer sizes were ranged over. The same strategy was used to find the values of the learning rate and momentum terms which gave best performance for each network type. The optimal size of the moving window of the TD-ELM for this task was set to the value of the delay plus the current time step (i.e. a length of two), while the input scale to the hidden layer and the hidden layer size were varied over using the same approach as for the ESN and R^2SP parameters. Table 6.1 shows the best parameters for each architecture, their corresponding errors and computation times (in hours:minutes:seconds) for the polynomial dataset[1]. The best network configurations for each network architecture are also shown in Table 6.1 below.

[1]All computation times reported in this thesis were obtained by simulating each architecture on a dual core machine with Intel ® Pentium ® D CPU 2.8 Ghz and 2 GB RAM.

Table 6.1: The average NRMSE test errors (with the standard deviations given in parentheses) obtained from an ESN architecture, R^2SP, an Elman network (EN), an EN trained with backpropagation through time (EN-BPTT), an EN trained with real-time recurrent learning (EN-RTRL), a Layer Recurrent Network (LRN), a distributed time-delay neural network (DTDNN) and a time-delayed extreme learning machine (TD-ELM) when applied to the polynomial dataset along with the computation times for training. Topological settings (i.e. the number of hidden layers and neurons in each layer) for each network are shown in the form *hl(nns)* where *hl* is the number of hidden layers and *nns* is the number of neurons in each layer. The best score is given in bold.

Type *hl(nns)*	**NRMSE (std-dev)**	**Input Scale**	**Spec Rad**	**Computation time (hrs:mins:secs)**
ESN 1(200)	$0.7368(9.399\times10^{-4})$	25.1	1.2	00:03:43
R^2SP 3(200)	$0.5241(9.5164\times10^{-4})$	2.5	0.6	00:06:47
TD-ELM 1(600)	$\mathbf{0.423(9.59\times10^{-4})}$	25.1	n/a	00:07:47
EN 1(10)	0.7805(0.1071)	n/a	n/a	23:37:20
LRN 2(10)	0.7750(0.1555)	n/a	n/a	61:50:24
DTDNN 2(10)	0.6523(0.0866)	n/a	n/a	49:46:12
EN-BPTT	$0.7697(4.47\times10^{-5})$	n/a	n/a	02:01:26

Continued on Next Page...

Table 6.1 – Continued

Type hl(nns)	**NRMSE (std-dev)**	**Input Scale**	**Spec Rad**	**Computation time (hrs:mins:secs)**
1(50)				
EN-RTRL 1(30)	0.7698 (4.47×10^{-5})	n/a	n/a	126:36:36

As Table 6.1 shows, on average the TD-ELM architecture (using a window length of two time steps) outperforms an ESN, R^{2}SP and the other traditional RANNs with the best TD-ELM scoring 0.4083 compared to a best ESN score of 0.7348 and a best R^{2}SP score of 0.5218. Using the Mann-Whitney U-test [334] for significance revealed that the errors between the R^{2}SP and the TD-ELM are significantly different (p value $2.56 \times10^{-34} < 0.05$), whilst the R^{2}SP offered a significant improvement in performance when compared to the ESN (p value $2.56 \times10^{-34} < 0.05$). All three architectures offered a significant improvement in performance in comparison to the EN, LRN and DTDNN architectures, with the exception of the ESN whose performance in comparison to the LRN was not significant. While this may be so, the small number of LRN networks trained makes this inconclusive. To validate this further, more LRN networks need to be trained in order to increase the number of networks' performance the ESN is compared against.

The Mann-Whitney U-test for significance was chosen over other tests for significance such as the *t*-test as the distribution of the test errors could not easily be verified as being normally distributed and the standard deviations between some errors were not

the same. The fact that the TD-ELM approach offers better performance than the ESN agrees with the results reported by Li et al [311] which were discussed previously in Chapter 4.

As a result of local minima, the average performance of the DTDNN, EN and LRN architectures is sub-optimal. In the case of the DTDNN, one network obtained an error score of 0.5005 which is slightly better than the R^2SP approach. However, as the table shows, on average the error of the DTDNN was higher than the R^2SP which was probably caused by local minima as suffered by the other four sub-optimal networks. It is conceded that better performance could have been obtained given enough time. However, the slow and complex training of these more conventional approaches made obtaining optimal performance very slow and difficult.

As R^2SP and TD-ELM have the advantage of fast and simple training, due to their similarities with standard RC and ELM approaches, they were also much faster to train than all of the other RANNs with the exception of the ESN. This was due to the added complexity of R^2SP and TD-ELM when compared to the ESN as a result of more neurons in the static layers of the R^2SP and the increased size of the input and hidden layers of the TD-ELM. However, the extra computation delivers a significant improvement in performance. As already discussed in Chapters 2 and 3, the traditional RANN training approaches used are slow and complex as is shown by their computation times. These times in some cases, such as the LRN being around 62 hours, are very long. These networks were simulated using the off-the-shelf Matlab Neural Network toolbox. Faster training times may be obtainable through the use of other software, as has been shown by the reduction in training time when using the Čerňanský RANN toolbox [320] for the BPTT training algorithm, although the RTRL algorithm from

the same toolbox was by far the most computationally expensive for this task. The Matlab Neural Network toolbox was chosen due to its wide applicability, ease of use and extendibility for these standard RANN training approaches.

Table 6.1 above also shows that the values of the input scaling and spectral radius that offered best performance for an ESN are quite high, which may have had a detrimental effect on the memory capacity of the reservoir of the ESN. This is not the case for R^{2}SP which has much smaller tuned values. Although the memory capacity, as defined by the Memory Capacity measure (MC), required from the task is small (i.e. one time step), the high values of input scale and spectral radius may also lead to an unstable ESN. In order to gain further insight into the characteristics of the best performing ESN and R^{2}SP networks as caused by their parameters shown in Table 6.1 the maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), MC and ASE of both architectures were calculated, and are shown in Table 6.2.

Table 6.2: The average maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviations from linearity (δ_ϕ), memory capacity (MC) and average state entropy (ASE) (standard deviations shown in parenthesis) of an ESN and an R^{2}SP using their tuned parameters for the 4^{th} order polynomial task.

Type	$\widetilde{LLE}_{max}$	$\delta_{\phi_{res}}$	$\delta_{\phi_{stat_1}}$	$\delta_{\phi_{stat_2}}$	$\delta_{\phi_{comb}}$	**MC**	**ASE**
ESN	0.1488	0.793	n/a	n/a	n/a	6.54	-1.08
	(0.0059)	(0.0024)				(0.13)	(0.0931)
R^{2}SP	-0.52	0.73	0.71	0.73	0.73	17.84	-2.8715
	(0.0055)	(0.0015)	(2.85×10^{-7})	(0.0012)	(0.0013)	(0.33)	(0.1081)

Inspection of Table 6.2 reveals a few interesting points. Firstly, the values of the maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$) of the ESN are above zero, which may be an indication of instability and could be a reason for its inferior performance. The $\widetilde{LLE}_{max}$ of the R^{2}SP-reservoir, however, is below zero which indicates that theR^{2}SP-reservoir is stable. The ESN also has a higher deviation from linearity than the R^{2}SP which include its R^{2}SP-reservoir, $stat_1$, $stat_2$ and all three layers combined. This is an indication that the ESN contains more non-linear reservoir neurons than any of the layers in the R^{2}SP, especially the R^{2}SP-reservoir. As a result of this the MC of the ESN is much lower than that of the R^{2}SP-reservoir which is also shown in Table 6.2. The causes of a potentially unstable reservoir with a higher deviation from linearity and a lower MC are the high values of the input scale and spectral radius. This results in inferior performance of the ESN when compared to the R^{2}SP and TD-ELM.

In comparison, the R^{2}SP-reservoir has a relatively small input scale, and a spectral radius which is in the usual range below unity. Here, its tuned input scale is slightly higher than unity, which may be an indication that the R^{2}SP-reservoir does in fact perform some non-linear transformation as little computation is required of it in terms of short-term memory. Recall that in this simple task, the network is only required to remember input from one time-step in the past. This dataset required both non-linear mapping and (very) short-term MC, which as the results show, an ESN alone cannot do as well. By separating these two requirements using R^{2}SP, its R^{2}SP-reservoir layer can be mainly tuned towards handling the MC part of the task as $stat_1$ performs the instantaneous non-linear transformation of the input data. The TD-ELM's significant improvement in performance for this may be a result of the R^{2}SP-reservoir's fading memory, which for a task which requires temporal representation for only two time steps, may actually hinder performance as inputs from further in the past may still be

present within the reservoir. The TD-ELM's memory, which only stretches back to the previous time step does not encounter this problem.

Inspection of the ASE values of both architectures' reservoirs shows that the ESN has a higher ASE than the R^{2}SP-reservoir which contradicts the advice of Ozturk and Principe [235] who recommended maximising the ASE for best performance, as a higher ASE value indicates a higher diversity in a network's responses. This may be surprising, as the R^{2}SP outperforms the ESN with a lower ASE value for its R^{2}SP-reservoir. One possible explanation for this is that the R^{2}SP-reservoir was tuned towards a larger MC as a result of its input scale and spectral radius values which enabled best performance. Therefore its state space is smaller in comparison to the ESN as less non-linear separation is required from its R^{2}SP-reservoir, hence a smaller ASE value. The ESN on the other hand, also had to perform the non-linear mapping of the input data and, as indicated by its higher valued parameters, its reservoir neurons operated in their more non-linear regions. As a result, the ESN mapped the input onto a higher dimensional state space than the R^{2}SP-reservoir, hence a higher ASE value obtained for the fourth order polynomial task using the ESN approach. As a result of this behaviour, a larger ASE value is expected to be obtained for the ESN when compared to the R^{2}SP-reservoir for the remainder of the case studies investigated, should the R^{2}SP-reservoir be tuned mainly towards MC as was the case here.

6.3.2 Extended polynomial

Several experiments were conducted for this task by incrementing over the degree of non-linearity within the polynomial and the amount of memory required from the task. As one of the main questions of this research concerned the memory and non-linearity present in an ESN and an R^{2}SP, only these two architectures were investigated when

applied to this dataset. As with the fourth order polynomial task an ESN reservoir size of 200 neurons was used, while each layer of the R^{2}SP contained 200 neurons giving a total of 600 neurons.

6.3.2.1 Increasing the power and delay of the polynomial

Initially, the performance of an ESN and R^{2}SP were compared when increasing the power (or non-linearity, denoted by p) and the delay (or memory requirements, denoted by d) simultaneously. Table 6.3 below shows the tuned parameters and errors of both architectures when increasing both p and d at the same time with the best test errors shown in bold. Figure 6.1, also below, shows a graph of both architectures' errors.

Table 6.3: The average test errors (standard deviation shown in parentheses) for the extended polynomial dataset, with corresponding input scale (ι) and spectral radius (ρ) parameters which gave best performance obtained from an ESN and an R^{2}SP when increasing the delay (d) and power (p), with the better scores shown in bold.

	ESN			**R^{2}SP**		
p **and** d	ι	ρ	**NRMSE**	ι	ρ	**NRMSE**
1	0.63	0.8	$0.46(8.23\times10^{-5})$	0.1	0.4	$\mathbf{0.029(2.86\times10^{-6})}$
2	0.25	0.6	$0.7(1.37\times10^{-4})$	0.1	0.2	$\mathbf{0.6196(2.26\times10^{-4})}$
3	0.63	0.8	$0.977(1.22\times10^{-4})$	0.1	0.6	$\mathbf{0.5875(6.4\times10^{-4})}$
4	0.25	0.8	$0.9711(2.84\times10^{-4})$	1.58	0.2	$\mathbf{0.4669(4.14\times10^{-4})}$
5	0.1	0.4	0.9305(0.009)	0.63	0.4	$\mathbf{0.486(5.82\times10^{-4})}$
6	0.25	0.4	0.9375(0.0011)	1.58	0.4	**0.5669(0.0011)**
7	1.58	1	$1.001(3.69\times10^{-4})$	0.25	0.6	**0.4111(0.0383)**
8	25.1	0.6	$1.0005(4.18\times10^{-4})$	0.63	0.4	**0.48(0.0447)**
9	3.98	0.8	$0.9976(6.62\times10^{-4})$	0.1	0.4	**0.4727(0.009)**
10	0.63	0.6	$0.9965(1.65\times10^{-4})$	0.1	0.4	**0.5906(0.0011)**
11	25.1	0.2	$1(1.96\times10^{-4})$	1.58	0.2	$\mathbf{0.5927(2.42\times10^{-4})}$
12	0.25	1.2	$1.0005(6.17\times10^{-4})$	0.25	0.4	$\mathbf{0.6702(4.1\times10^{-4})}$
13	3.98	0.2	$0.9999(7.89\times10^{-5})$	0.25	0.6	$\mathbf{0.6782(5.61\times10^{-4})}$
14	3.98	1.2	$1.0007(4.74\times10^{-4})$	0.1	0.8	$\mathbf{0.7067(3.83\times10^{-4})}$
15	0.63	0.8	$0.9851(5.61\times10^{-4})$	0.1	0.8	$\mathbf{0.7929(5.04\times10^{-4})}$
20	0.25	0.8	$0.9926(7.36\times10^{-4})$	0.1	0.8	$\mathbf{0.7771(7.73\times10^{-4})}$
50	1.58	1.2	$1.0011(4.94\times10^{-4})$	0.25	0.4	$\mathbf{0.78(4.67\times10^{-4})}$
100	0.1	1.2	$1.0005(6.12\times10^{-4})$	0.1	0.8	$\mathbf{0.8358(3.59\times10^{-4})}$
150	0.25	1.4	$1.0006(4.09\times10^{-4})$	0.25	0.6	**0.8891(0.0017)**

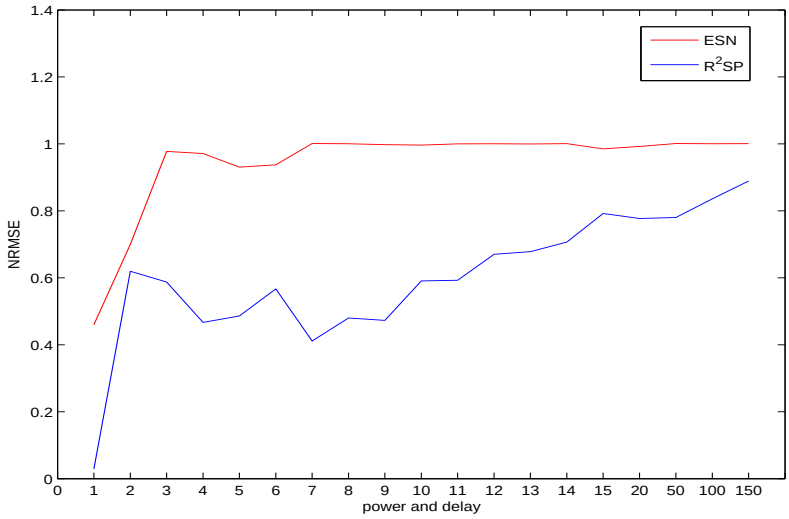


Figure 6.1: The NRMSE error of an ESN (red) and an R^{2}SP (blue) obtained when increasing the power and delay simultaneously for the extended polynomial task. Note that the scale along the X axis is non-linear beyond 15.

As Table 6.3 and Figure 6.1 show, the R^{2}SP network outperforms an ESN when applied to the extended polynomial task over all values of the power and delay. This difference was confirmed as significant using the Mann-Whitney U-test with a 95% confidence level for all cases. As the values of p and d increase, the ESN's error quickly approaches a value around unity (a NMRSE value of unity indicates that a network is essentially guessing). In contrast, the error of R^{2}SP never reaches an NRSME of unity, and for the majority of p and d values its error is approximately half that of the ESN. This remains the case up until the power and delay values are greater than nine, where the task becomes more difficult as both networks are required to recall inputs further into the past and separate data into a more non-linear output. As the values of p and d increase further the performance of the R^{2}SP becomes gradually worse and it is expected to reach a similar level to an ESN when values of p and d exceed 150. As

the number of coefficients increases dramatically with each increment in p, a p value greater than 150 was not investigated in this work.

An interesting observation can be made by inspection of Figure 6.1 when the delay and power are equal to unity. The R^{2}SP network has a very small error, whereas the error of an ESN is considerably higher. To investigate why this may be the case, the delays (the time for each layer to react to its input) present in each layer of R^{2}SP were analysed. This involved feeding an R^{2}SP network with an input of -1 for 50 time steps, followed by an input of +1 for 50 time steps and analysing the activations of the R^{2}SP-reservoir, $stat_1$ and $stat_2$ layers of the R^{2}SP to assess their reaction to a change in input (for illustration purposes, a simple network where each layer contained five neurons was investigated). The activations of each layer of the R^{2}SP are shown in Figure 6.2 below.

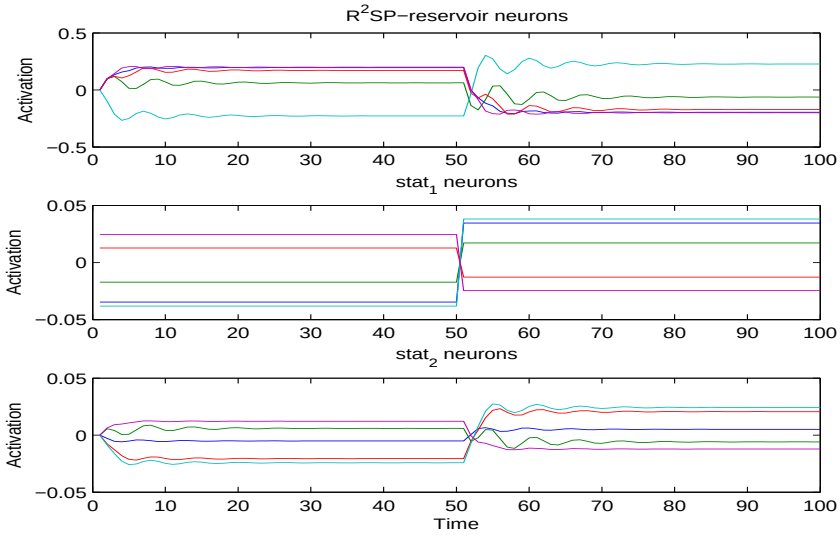


Figure 6.2: The activations of five neurons in each layer of an R^{2}SP architecture shown by the different colour plots. The middle plot shows that $stat_1$ has an instantaneous reaction to the change in the input data from -1 to + 1 which occurred at $t = 51$. In the case of the R^{2}SP-reservoir and $stat_2$ neurons (top and bottom plots respectively), some time is required for their neurons' activations to adjust to the change in input before their activations reach a stable state, which is a result of the recurrent connections present within the R^{2}SP-reservoir. As $stat_2$'s inputs are the R^{2}SP-reservoir activations, its neuron's activations follow a similar pattern of activation to those of the R^{2}SP-reservoir neurons. This shows that reservoirs are quite slow in reacting to a change in input. In scenarios with fast changing highly non-linear data present a fast reaction to a change of input may be necessary for improved performance.

Figure 6.2 offers one explanation as to why the R^{2}SP offers improved performance even with low values of p and d in the extended polynomial task and also in the case of the fourth order polynomial investigated in the previous section. The R^{2}SP-reservoir layer alone takes some time (in Figure 6.2 shown above approximately 20 time steps) before settling to a stable state when fed with a constant input as a result of its

recurrent connections which cause previous activations to interfere with the current neuron activations. The neurons of the R^{2}SP's $stat_1$ layer, which have no recurrent connections, react instantaneously to the change in input which make them well suited to the ever changing input data of the extended polynomial task, hence the R^{2}SP's improvement in performance. This instantaneous reaction enables the neurons of $stat_1$ to perform the non-linear transformation of the input data, while the R^{2}SP-reservoir performs the memory mapping of the input as a result of its recurrent connections and its slower changing dynamics.

Inspection of the tuned values for the input scale of the ESN in Table 6.3 above reveals that overall it generally prioritises MC over non-linear separation as indicated by the majority of values being lower than unity, although some higher values can also be observed in cases where p and d are equal to 8, for example. In two cases, the tuned input scale of the ESN has a value of 25.1, which is very high and is expected to have a detrimental effect on the MC of the ESN. The fact that the best performing ESN's performance was mostly geared towards maximising MC was also found when applying a reservoir to the extended XOR task [226] as indicated by its tuned parameters. In comparison, the input scale to the R^{2}SP-reservoir never has a value as high as that of the ESN. This should allow the R^{2}SP-reservoir to possess a higher MC.

Inspection of the spectral radius for the ESN shows that the tuned value is again generally below unity, indicating that the ESN's reservoir prefers MC over non-linearity, although in scenarios where p and d are greater than 11 the spectral radius does exceed unity which may reduce the MC of the ESN, especially in the cases where p and d are at their highest values. This does not necessarily mean, however, that the ESN is unstable. In order to analyse these results further and to prove the aforementioned

hypotheses of the dynamics of both architectures' reservoirs, the MC and deviation from linearity for both architectures using the tuned input scale and spectral radius parameters for each value of p and d (from Table 6.3) are plotted below in Figures 6.3 and 6.4 respectively.

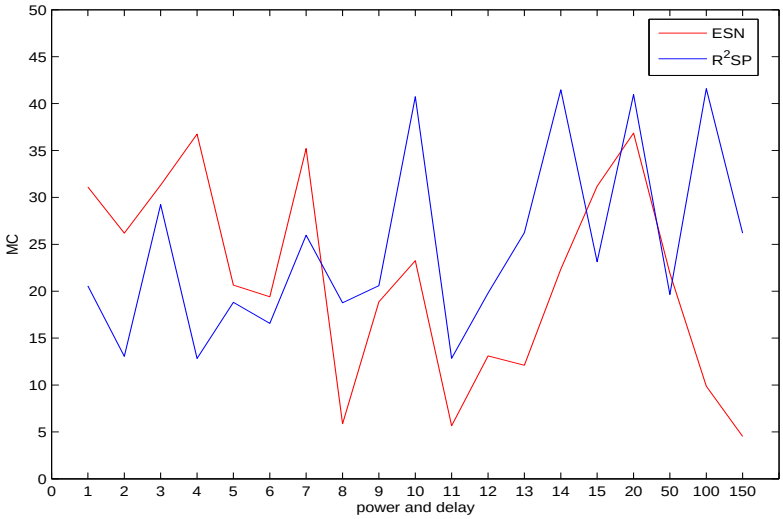


Figure 6.3: The memory capacity (MC) of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the power and delay for the extended polynomial task. Note that the X axis is non-linear after power and delay values exceed 15. This shows that for lower values of p and d (below seven) the ESN has a higher MC than the R^{2}SP's. As p and d increase, the MC of the ESN is generally lower than the R^{2}SP's as a result of its input scale and spectral radius values which allowed best performance. For high values of p and d equal to 100 and 150, the high valued parameters for the ESN have a detrimental effect on its MC, where, for this task, it is required the most.

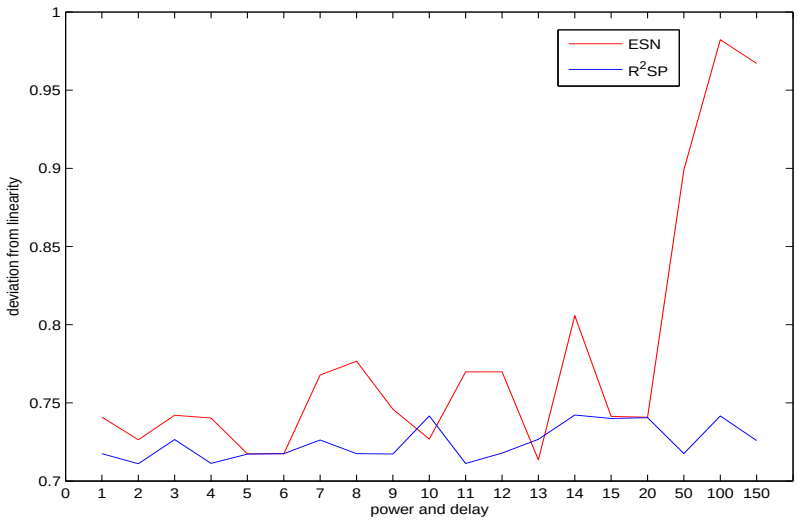


Figure 6.4: The deviation from linearity of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the power and delay for the extended polynomial task. Note again the non-linear increase in the power and delay values above 15. This shows that the ESN's deviation from linearity is generally higher than the R^{2}SP's, especially where p and d are equal to 100 and 150 where higher amounts of non-linear separation are required, but as a result cause the ESN's MC to decrease as shown above in Figure 6.3.

Figure 6.3 shows that the ESN approach begins with a higher MC than that of the R^{2}SP up until p and d exceed a value of eight, after which the R^{2}SP has a higher MC for the majority of p and d increments. As a delay of eight time steps is not too far into the past, it can be argued that the smaller MC of the R^{2}SP up until this point does not affect its performance. The MC of both networks becomes more important as the value of d increases. The figure shows that in all but two cases where p and d are higher than eight (where p and d equal 15 and 50) the R^{2}SP-reservoir has a higher MC, especially towards the higher values of p and d. One can argue that the decrease in the MC of an ESN at these extremes account for its increased error rates as shown

in Table 6.3 and Figure 6.3. This decrease in MC of the ESN is due mainly to large values of input scale when p and d are equal to 8 and 11 and large values of its spectral radius above unity when p and d become greater than 20, as was also shown in Table 6.3.

As a result of the increase in spectral radius after p and d exceed a value of 20, the ESN's deviation from linearity also increases as is shown in Figure 6.4. For the majority of increments in p and d the deviation from linearity of the ESN was close to the reservoir of the R^{2}SP with the exception of when it had large values for its tuned input scale when p and d were equal to 8 and 11. Once p and d become larger than 20, the spectral radius of the ESN becomes higher than unity which is most likely to be caused by the increased non-linearity of the dataset.

In summary, this subtask of simultaneously increasing the power and delay of the extended polynomial task has shown that with increasing memory and non-linearity in a dataset an ESN is unable to perform as well as the R^{2}SP architecture as a result of a trade-off between its capability to recall inputs that stretch back into the past and to separate highly non-linear data at the current time step. With requirements for high values of memory and non-linear separation the best values of the ESN's parameters, in particular its spectral radius, become high. As a result of these high valued parameters the MC of an ESN decreases dramatically, while the non-linearities present inside the ESN increase as shown by an increase in its deviation from linearity. Separating these two requirements of the task by adopting the R^{2}SP approach allows the R^{2}SP-reservoir layer to perform the memory part of the task, while the non-linear transformation is performed by the memoryless feedforward layer. As the R^{2}SP's $stat_1$ layer receives the same input as the R^{2}SP-reservoir, it performs an instant non-linear transformation of

the input which, in a task where the input is changing constantly, aids performance also. This is something that an ESN is unable to perform well as a result of its recurrent connections which interfere with the neurons' activation at the current time step which was also shown in this task.

6.3.2.2 Increasing the delay of the polynomial

The performance of both architectures was further compared on the same extended polynomial by increasing the delay of the task, d, whilst keeping the non-linearity of the task, p, equal to unity. This was to investigate the capabilities of the R^2SP approach and test the hypothesis that by having additional non-linear layers its R^2SP-reservoir has an increased MC as a result of its tuned parameters. The tuned parameter values and NRMSE errors of an ESN and R^2SP are shown in Table 6.4 below, while the errors of both architectures are plotted in Figure 6.5.

Table 6.4: The average test errors in NRMSE (standard deviation shown in parentheses) for the extended polynomial dataset, with corresponding tuned input scale (ι) and spectral radius (ρ) parameters obtained from an ESN and an R^{2}SP when increasing the delay (d) and keeping the power (p) equal to unity, with the better errors shown in bold.

	ESN			**R^{2}SP**		
d	ι	ρ	**NRMSE (std-dev)**	ι	ρ	**NRMSE (std-dev)**
2	1.58	0.4	$0.4531(1.27\times10^{-4})$	0.25	0.8	$\mathbf{0.0405(5.2277\times10^{-6})}$
3	0.63	0.2	$0.4537(6.5121\times10^{-5})$	0.63	0.4	$\mathbf{0.0502(3.3234\times10^{-5})}$
4	0.1	0.4	$0.4548(5.1364\times10^{-5})$	1.58	0.2	$\mathbf{0.0559(3.0297\times10^{-5})}$
5	0.25	0.4	$0.4538(8.3421\times10^{-5})$	0.25	0.4	$\mathbf{0.0643(1.9134\times10^{-5})}$
6	0.1	0.2	$0.4572(6.1802\times10^{-5})$	0.63	0.8	$\mathbf{0.0708(4.1363\times10^{-5})}$
7	0.1	1	$0.4605(3.265\times10^{-4})$	0.25	0.8	$\mathbf{0.0748(2.77\times10^{-5})}$
8	0.1	0.4	$0.4602(7.2924\times10^{-5})$	0.1	0.8	$\mathbf{0.0822(1.1511\times10^{-5})}$
9	0.63	0.6	$0.4587(1.7542\times10^{-4})$	0.25	0.8	$\mathbf{0.0835(7.97\times10^{-5})}$
10	0.63	0.4	$0.4587(2.7628\times10^{-4})$	0.1	0.4	$\mathbf{0.0899(3.2504\times10^{-5})}$
11	0.25	0.4	$0.4631(2.2495\times10^{-4})$	0.25	0.8	$\mathbf{0.0917(2.5336\times10^{-5})}$
12	0.1	0.4	$0.462(2.6763\times10^{-4})$	0.1	0.4	$\mathbf{0.0924(4.5029\times10^{-5})}$
13	0.1	0.8	$0.4625(1.9709\times10^{-4})$	0.63	0.6	$\mathbf{0.0956(8.909\times10^{-5})}$
14	0.25	0.8	$0.4602(1.8994\times10^{-4})$	0.1	0.6	$\mathbf{0.0975(3.8529\times10^{-5})}$
15	0.25	0.8	$0.4617(1.7078\times10^{-4})$	0.1	0.8	$\mathbf{0.1013(3.4059\times10^{-5})}$
20	0.25	0.8	$0.466(5.0056\times10^{-4})$	0.1	1	**0.1241(0.0013)**
50	0.1	1	0.6574(0.0582)	0.1	1	**0.4622(0.0722)**
100	0.1	1	0.9935(0.0061)	0.1	1	**0.8809(0.0074)**

Continued on Next Page...

Table 6.4 – Continued

	ESN			**R^{2}SP**		
d	ι	ρ	**NRMSE (std-dev)**	ι	ρ	**NRMSE (std-dev)**
150	0.1	1	0.9986(0.0011)	0.1	1	**0.8918(0.0014)**

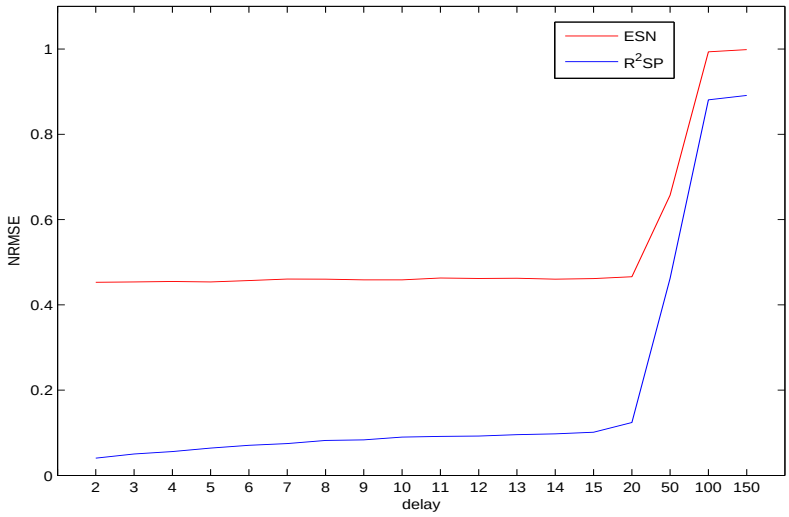


Figure 6.5: The NRMSE of an ESN (red) and an R^{2}SP (blue) when increasing the delay for the extended polynomial task. Note that the sharp increase in error when the delay is greater than 15 is partly due to the non-linear increase in the delay value scale beyond 20.

As Table 6.4 and Figure 6.5 show, the R^{2}SP approach outperforms an ESN over all iterations of delay (d). Significance testing using a Mann-Whitney U-test revealed that the R^{2}SP's improved performance was indeed significant for all values of d (p value $<$ 5%). Inspection of the tuned parameters reveals that both architectures are tuned

towards maximising their MC as no values above unity are observed. As this task requires a high MC alone, this is expected. As the delay increases, the error of both architectures increases steadily up until d is greater than 20, where a large increase in error is observed for both architectures which is probably due to the large increase in the delay from 20 time steps to 50 time steps. Had delays between these two values been investigated, the increase in error would be less extreme. As was the case when increasing the values of both p and d, the gain in performance of the R^{2}SP could be attributed to the fact that the neurons of its $stat_1$ layer react instantaneously to a change in the input. In this situation the change of the target output is not as drastic as when increasing the power of the polynomial, as a 1^{st} order polynomial was used (p is equal to unity), but even this small amount of instant non-linear transformation is difficult for an ESN to possess as well as a high MC, so this may be a factor in its inferior performance. As before, the MC and non-linearities contained in both architectures were calculated using the values of their respective input scale and spectral radius which enabled best performance for each value of d, and are shown in Figures 6.6 and 6.7 below.

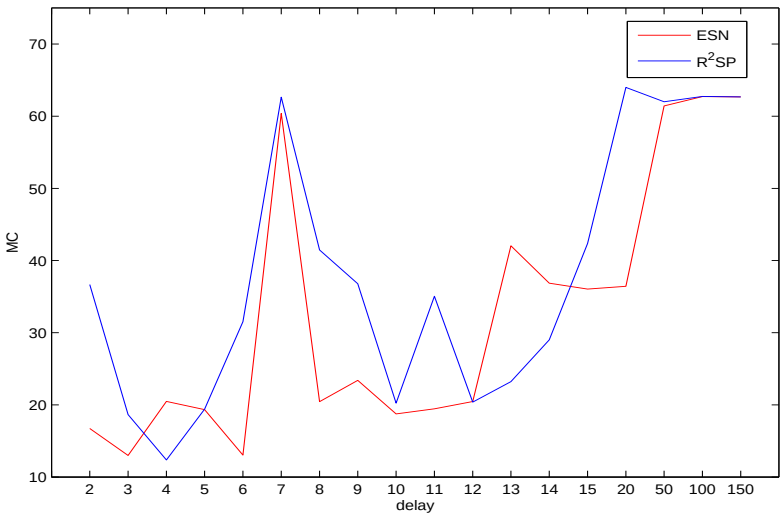


Figure 6.6: The memory capacity (MC) of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the delay for the extended polynomial task. This shows that the MC of the two architectures is quite similar throughout the increments of d, which is expected as the increase in MC was a requirement of the task. The R^{2}SP-reservoir has a slightly higher MC for most increments of d however, which could be a factor in its improvement in performance. Apart from when d is equal to seven, the MC of both architectures generally increases as d increases, which is also expected.

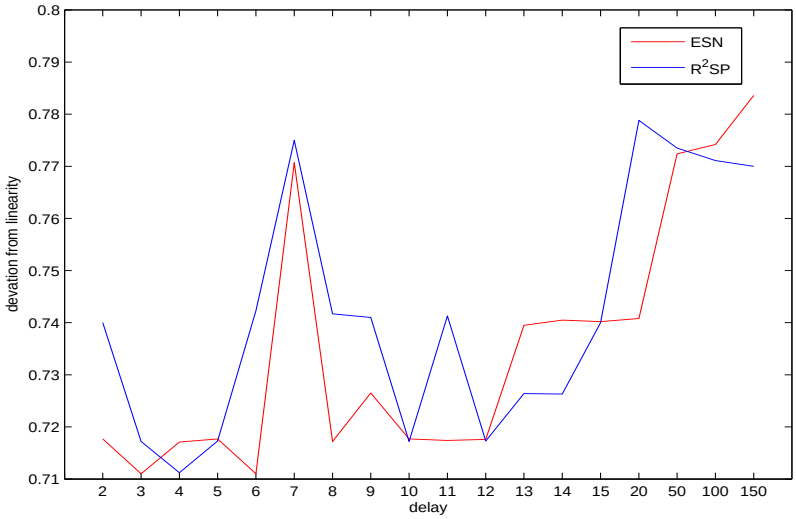


Figure 6.7: The deviation from linearity of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the delay for the extended polynomial task. As with the MC plotted above in Figure 6.6, the deviation from linearity of the two reservoirs is similar. Here the R^{2}SP generally has a higher deviation from linearity, but as Figure 6.6 showed, the relatively small increases in its deviation from linearity do not have a detrimental effect on its MC. Note that the deviation from linearity for both architectures here never becomes as large as the case when increasing p and d simultaneously, as shown in Figure 6.4 above, hence higher values of non-linearity when increasing d give higher values of MC in this case.

Figure 6.6 shows that the MC of the two architectures is quite similar, which is expected as the task required both architectures to have a high MC, especially as the delays increased. The R^{2}SP-reservoir has a higher MC when d is equal to 2 which could explain the large difference in performance even at this point. In these cases the increased MC of the reservoir inside the R^{2}SP may be a result of the small amount of non-linear separation being performed by $stat_1$ rather than the reservoir, as is the case of the ESN which as a result will decrease its MC. However, the difference in

performance is similar throughout for the other values of d, which may suggest that the instantaneous non-linear transformation of $stat_1$ may also aid in performance here, as was shown previously in section 6.3.2.1. Where d is equal to 20, the MC of the R^{2}SP's reservoir is lower than the ESN's, which is also reflected in the increase in error of both architectures: the error of the R^{2}SP increases by 0.02, while the error of the ESN increases by a smaller 0.0043. Despite this, the error of the R^{2}SP is still much less than that of the ESN. Figure 6.6 also shows that the MC of the two architectures is very similar when d is equal to 100 and 150, where a high MC is required.

Figure 6.7 shows that the deviation from linearity of both architectures follows a similar pattern to their MCs. The R^{2}SP has a higher deviation from linearity for the majority of the increments of d, with noticeable exceptions where d is equal to 100 and 150. The higher values of deviation from linearity at these points may explain the increased error of the ESN at these points, but the values of the deviation from linearity for both architectures do not change as drastically when increasing both p and d as shown previously in Figure 6.4, where the deviation from linearity values often exceeded 0.8. The changes here in the deviation from linearity are, as always, due to the changes in the values of the network parameters, which in this case are tuned towards maximising MC. As Table 6.4 shows, the tuned input scale and spectral radius parameters for both architectures generally stay below unity in order to maximise their MCs. In this case, a higher deviation from linearity as caused by a spectral radius value close or equal to unity cannot be seen as a bad characteristic of the network, as a slight increase in the deviation from linearity is also reflected in higher MCs. However, once the neurons' working points move even further away from their linear region as a result of the network's parameters (in this case the input scale and spectral radius), larger increases in the deviation from linearity are observed, and as a result, the MC of the

network then decreases as was shown previously in Figures 3.14 and 3.15 of Chapter 3.

6.3.2.3 Increasing the power of the polynomial

As a final comparison for this dataset, the performance of both architectures was investigated when increasing the non-linearity of the output, p, and leaving the delay, d, set at unity. This comparison was conducted also with aim of analysing the amount of non-linear transformation performed by the R^{2}SP-reservoir: if the feedforward memory-less layers performed the majority of the non-linear separation, then with highly non-linear data its R^{2}SP-reservoir should still be tuned towards maximal MC. In the case of the ESN, its neurons will become more non-linear as is required by the task. The parameters which enabled best performance for both architectures and their errors when iterating over p are shown in Table 6.5 below.

Table 6.5: The average NRMSE test errors (standard deviation shown in parentheses) for the extended polynomial dataset, with corresponding tuned input scale (ι) and spectral radius (ρ) parameters obtained from an ESN and an R^{2}SP when keeping the delay (d) set to unity and incrementing the power (p), with the best scores highlighted in bold.

	ESN			**R^{2}SP**		
p	ι	ρ	**NRMSE (std-dev)**	ι	ρ	**NRMSE (std-dev)**
2	1.58	0.6	$0.7102(1.6806\times10^{-4})$	0.25	0.4	$\mathbf{0.6184(2.0594\times10^{-4})}$
3	63.09	0.8	$0.974(1.8612\times10^{-4})$	0.1	0.2	$\mathbf{0.5879(1.9535\times10^{-4})}$
4	3.98	0.4	$0.9696(2.6611\times10^{-4})$	0.1	0.2	$\mathbf{0.464(1.933\times10^{-4})}$
5	25.11	1.4	0.9443(0.0014)	1.58	0.2	$\mathbf{0.4736(3.9854\times10^{-4})}$
6	25.11	1.2	$0.9495(9.9996\times10^{-4})$	0.1	0.4	**0.5558(0.0011)**
7	25.11	0.2	$0.9988(4.2774\times10^{-4})$	0.1	0.2	**0.3939(0.0203)**
8	25.11	1.2	$0.9979(4.663\times10^{-4})$	0.25	0.2	$\mathbf{0.4686(2.9251\times10^{-4})}$
9	63.09	0.6	$0.9961(2.3224\times10^{-4})$	0.1	0.4	$\mathbf{0.4754(2.85\times10^{-4})}$
10	25.11	0.8	$0.9968(4.0642\times10^{-4})$	0.1	0.2	$\mathbf{0.5955(1.5872\times10^{-4})}$
11	25.11	1	$0.9967(4.101\times10^{-4})$	0.63	0.2	$\mathbf{0.5886(5.8998\times10^{-4})}$
12	25.11	1.2	$0.9971(4.805\times10^{-4})$	0.63	0.2	$\mathbf{0.6665(5.8707\times10^{-4})}$
13	25.11	1	$0.9945(4.2306\times10^{-4})$	3.98	0.2	$\mathbf{0.6747(5.3391\times10^{-4})}$
14	25.11	1.2	$0.9944(4.2398\times10^{-4})$	0.1	0.6	$\mathbf{0.6979(7.7577\times10^{-4})}$
15	25.11	1.2	0.9659(0.0013)	3.98	0.2	$\mathbf{0.7318(8.2124\times10^{-4})}$
20	25.11	1	0.9747(0.0012)	1.58	0.2	$\mathbf{0.7413(9.0894\times10^{-4})}$
50	25.11	0.8	$0.999(5.6989\times10^{-4})$	0.63	0.2	**0.8061(0.0013)**
100	25.11	0.6	$0.9999(3.1056\times10^{-4})$	0.1	0.2	$\mathbf{0.8254(5.6665\times10^{-4})}$

Continued on Next Page...

Table 6.5 – Continued

	ESN			**R^{2}SP**		
p	ι	ρ	**NRMSE (std-dev)**	ι	ρ	**NRMSE (std-dev)**
150	25.11	0.8	$0.9993(5.5492\times10^{-4})$	0.25	0.6	$\mathbf{0.8735(5.8929\times10^{-4})}$

As Table 6.5 shows, the R^{2}SP again offers superior performance compared to an ESN, which was found to be significant for each value of p and d using the Mann-Whitney U-test with a 95% confidence interval. With the exception of when p is equal to 2, the NRSME of the ESN is around unity, which as, mentioned previously in this section, means that the network is essentially guessing. The NRMSE of the R^{2}SP network never reaches unity however, although it does approach the same level of error with high values of p, where it is eventually expected to reach the same level of error as the ESN with increasing values of p which make the task more difficult. In comparison to the task when increasing the delay only, the NRSME of the two architectures becomes much higher for smaller values of p compared to the same d values. From this, one can conclude that for the extended polynomial task, non-linear separation as defined by the order of the polynomial may be a more difficult task than recalling inputs from the past. Figure 6.8 shows the NRSME of the two architectures, while the input scale values which enabled best performance are shown in Figure 6.9.

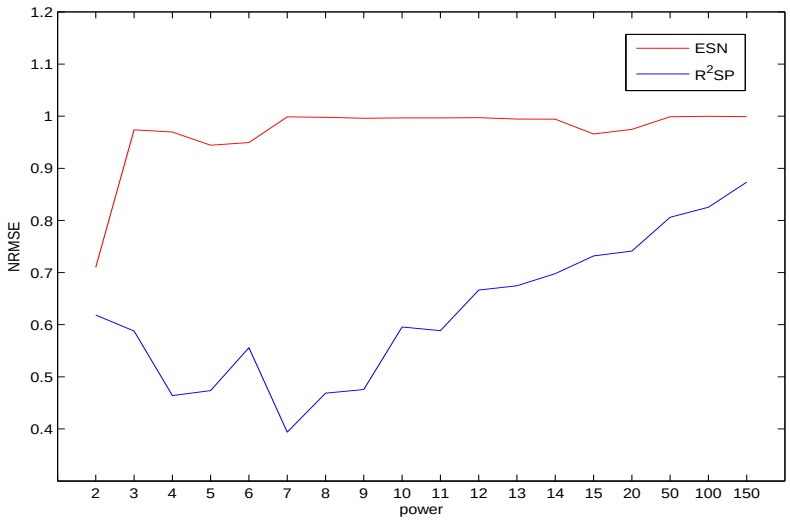


Figure 6.8: The test NRMSE of an ESN (red) and an R^2SP (blue) when increasing the power for the extended polynomial task showing the improved performance offered by the R^2SP. As the power increases, the error of the R^2SP increases also, where eventually with a p value higher than 150 it is expected to match the error of the ESN. The error of the R^2SP after p values greater than 15 increases dramatically as a result of the non-linear increase in the value of p after 20.

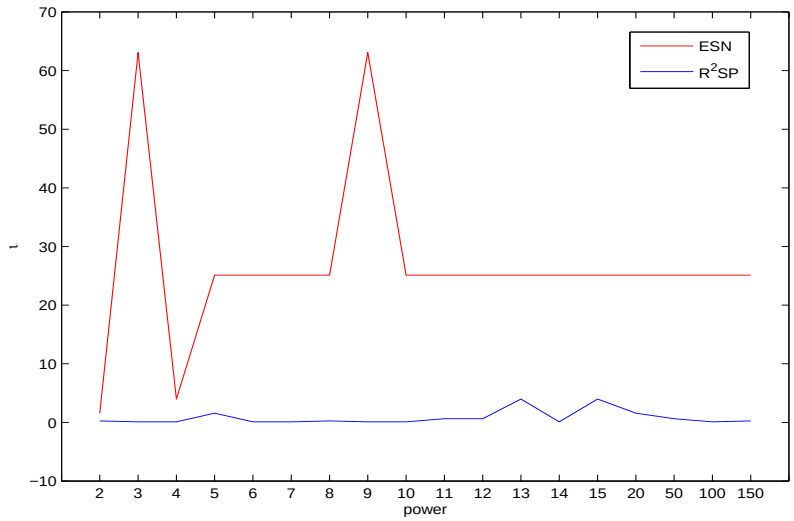


Figure 6.9: The tuned input scale of an ESN (red) and an R^{2}SP (blue) when increasing the power for the extended polynomial task. This shows that the input scale of the ESN is much higher than the input scale to the R^{2}SP-reservoir. As the task contains little MC requirement, this can be expected for the ESN as it is tuned towards maximal non-linear separation in order to achieve the best performance. As the non-linear layer $stat_1$ of the R^{2}SP performs this task, its R^{2}SP-reservoir is still tuned towards maximal MC, as shown by its smaller input scale values.

Figure 6.9 shows that the tuned input scale of the ESN is much higher than the input scale to the R^{2}SP-reservoir. For the ESN, this can be expected: as the target output data becomes more non-linear, a high input scale provides the best performance as it moves the working points of the reservoir neurons towards their non-linear regions where they transform the input onto a high dimensional state space in order to deliver better performance. The input scale to the R^{2}SP-reservoir on the other hand never becomes as high as the input scale of the ESN, confirming that the R^{2}SP-reservoir is required to perform little non-linear transformation as this is mainly taken care of by

$stat_1$.

The tuned spectral radius for the ESN is also high for some values of p. However, as a result of its tuned high input scale at these values, the effect of the spectral radius on the dynamics of its reservoir is less pronounced as was discussed earlier in Chapter 3. Conversely, the values of the tuned spectral radius for the R^2SP-reservoir stay below unity for all values of p. While a spectral radius around unity often gives the highest MC, a high MC for this task is not required as only one time step in the past needs to be recalled which may explain the many small values of spectral radius around 0.2 and 0.4. As a result of the high input scale to the ESN, it is expected that its MC will be low whereas its deviation from linearity will be high. The opposite is expected for the R^2SP-reservoir. The values of both of these measures are plotted in Figures 6.10 and 6.11.

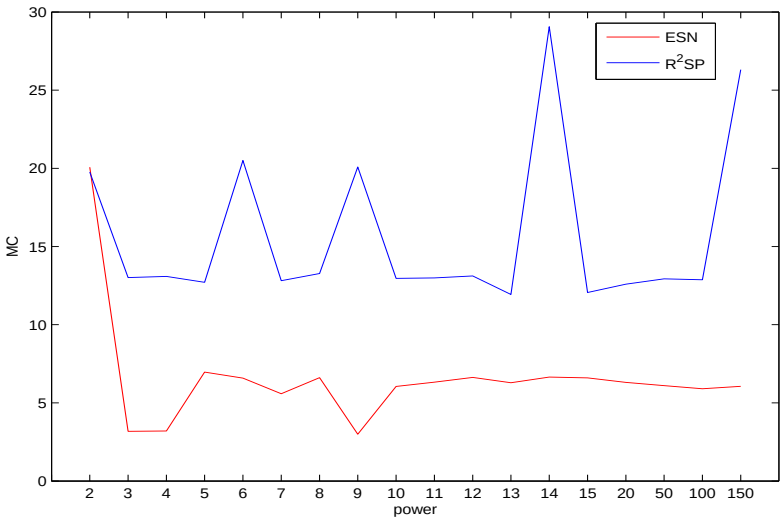


Figure 6.10: The memory capacity (MC) of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the power for the extended polynomial task. As a result of their tuned input scale and spectral radius, this shows that the R^{2}SP-reservoir has a much higher MC than the ESN. While little MC is required for this task, it shows that the majority of the non-linear transformation in the R^{2}SP architecture is performed by $stat_1$ rather than the R^{2}SP-reservoir as is the case when using an ESN.

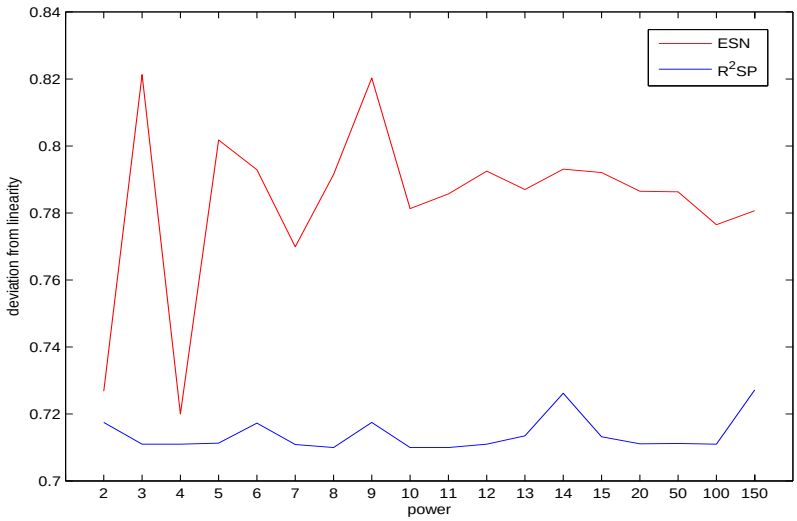


Figure 6.11: The deviation from linearity of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the power for the extended polynomial task which shows the ESN contains more non-linearities than the R^{2}SP-reservoir. While this is desirable for this task as the target data becomes increasingly non-linear with higher p values, it shows that the R^{2}SP allows its R^{2}SP-reservoir to be tuned towards maximum MC (as shown in Figure 6.10 above), as the majority of the non-linear transformation is performed by $stat_1$.

As Figure 6.10 shows, the MC of the R^{2}SP-reservoir is much higher than that of the ESN. Conversely, Figure 6.11 shows that the ESN has a higher deviation from linearity than the R^{2}SP-reservoir. These two observations are caused mainly by the tuned high input scales of the ESN when increasing the non-linearity of the dataset. Given the task, this is expected as little MC is required, whereas high amounts of non-linear transformation of the input data are required. By increasing the non-linearity of the dataset it shows that the non-linear transformation of the input is mainly performed by $stat_1$ of the R^{2}SP architecture which enables its R^{2}SP-reservoir layer to be tuned towards the maximum MC possible. This can be beneficial with tasks which require

both high MC and non-linear transformation where it is difficult to obtain a single reservoir which has high amounts of MC and non-linear separation due to the antagonistic trade-off between the two capabilities. As a result, their is no freedom in its parameter space to optimise its non-linear mapping capabilities [226]. As has been shown when increasing p and d simultaneously in this task and as can often be the case when dealing with real-world time-series data which require high amounts of MC and non-linear separation, by separating these two requirements using an approach such as R^{2}SP improved performance can be achieved. This may provide improved performance when compared to an ESN when applied to the EMAD concrete datasets, which contain high degrees of non-linearity and require memory of previous inputs.

6.3.3 Extended delayed XOR task

Similar to the extended polynomial dataset this task also allowed the non-linearity and memory required from the task to be varied. As with the extended polynomial task, only the ESN and R^{2}SP architectures were investigated for this task. Adopting the same approach as used by Verstraeten et al [226], both the power (or amount of non-linearity, denoted by p) and the delay (the amount of MC required, denoted by d) parameters were increased until a value of 10 was reached, after which only d was increased until a value of 15 was reached. As with the study by Verstraeten at el [226], a β (the bias scalar) value of 0.1 was found to aid performance for both architectures when ranging over the input scale and spectral radius. Using a small bias was shown to enable a reservoir to contain a level of asymmetric non-linearity required from the task through the product terms in the target output. Layer sizes larger than 200 neurons were not found to greatly impact performance of the ESN and R^{2}SP, therefore 200 neurons were used in each layer of the R^{2}SP and 200 for the reservoir of the

ESN. Table 6.6 shows the performance of both architectures along with their tuned parameters.

Table 6.6: The average NRMSE test errors, tuned input scale (ι) and spectral radius (ρ) parameters and Mann-Whitney U-test `p` values obtained from an ESN and an R^{2}SP when increasing the delay (d) and power (p) of the extended XOR dataset (standard deviation of the errors are shown in parentheses). The best performance for each increment of d and p are shown in bold as are the U-test `p` values which are lower than 0.05 and indicate a significant difference.

		ESN			**R^{2}SP**			
p	d	ι	ρ	**NRMSE (std-dev)**	ι	ρ	**NRMSE (std-dev)**	**Mann-Whitney `p` value**
1	1	1.58	0.4	0.0489 (0.0023)	0.1	0.6	**0.0479 (0.0044)**	**1.31×10^{-5}**
2	2	10	0.2	0.1348 (0.0091)	10	0.2	**0.1111 (0.0122)**	**3.88×10^{-27}**
3	3	3.98	0.2	0.3402 (0.0334)	3.98	0.2	**0.313 (0.0419)**	**1.93×10^{-8}**
4	4	3.98	0.2	0.663 (0.0176)	3.98	0.4	**0.6578 (0.0205)**	0.1891
5	5	0.1	0.4	**0.7572 (0.0033)**	0.1	0.6	0.7594 (0.003)	**2.44×10^{-7}**
6	6	0.25	0.6	**0.8187 (0.0035)**	0.25	0.6	0.8193 (0.0044)	**0.0164**

Continued on Next Page...

Table 6.6 – Continued

		ESN			**R^{2}SP**			
p	d	ι	ρ	**NRMSE (std-dev)**	ι	ρ	**NRMSE (std-dev)**	**Mann-Whitney p value**
7	7	0.25	0.4	0.8713 (0.0089)	0.63	0.4	**0.869 (0.0081)**	**0.0202**
8	8	0.63	0.4	0.9318 (0.0245)	0.63	0.8	**0.9066 (0.0047)**	**1.17×10^{-24}**
9	9	0.63	0.8	0.9395 (0.0064)	0.63	0.8	**0.9371 (0.0061)**	**0.0025**
10	10	0.63	0.8	0.9663 (0.0068)	0.63	0.8	**0.9646 (0.0065)**	0.1045
10	11	0.25	1	0.9829 (0.0063)	0.63	1	**0.972 (0.0071)**	**1.83×10^{-20}**
10	12	0.63	1	0.9827 (0.0039)	0.63	1	**0.9783 (0.007)**	**5.03×10^{-7}**
10	13	0.63	1	0.9875 (0.0051)	0.25	1	**0.985 (0.0067)**	**0.0027**
10	14	1.58	1.2	0.9987 (0.0027)	0.63	1	**0.9905 (0.0065)**	**1.6976×10^{-19}**
10	15	1.58	1	0.9979 (0.0024)	0.63	1	**0.9959 (0.0048)**	**0.0084**

As Table 6.6 shows, the two architectures score very similarly, with the R^2SP offering marginal improvement in 13 out of the 15 values of p and d. As with the previous results, the Mann-Whitney U-test was performed where this marginal improvement of the R^2SP was proven to be significant in 11 out of the 13 cases for `p` values less than 5% which are shown in the right-hand column of Table 6.6. In the remaining two cases where the R^2SP outperformed an ESN (p and d equal to four and ten) no significant difference between their errors was found. In the two cases where the ESN offered improved performance, with p and d values of five and six, it was found to outperform the R^2SP significantly using the same confidence level as above. While the improvement in performance of the R^2SP is not as pronounced as with the previous tasks, it does offer significantly improved performance for 11 out of the 15 different values of p and d, especially where the task becomes more difficult with p and d values greater than 10. The drop in performance of the R^2SP when compared to its performance so far could be down to the difficulty of the task since, as was shown in Chapter 5, it becomes very difficult after only a few increments of p and d.

Table 6.6 also shows that, as a result of their tuned parameters, both architectures are generally tuned towards maximising their MC. For low values of p and d, where they both equal two, three and five, both architectures have a high input scale. As memory of only a few time steps is required here the best performing reservoir would appear to be one which is tuned towards non-linear separation. However, as the values of p and d increase, the reservoir of the R^2SP and ESN are mostly tuned towards maximal MC, as shown by low input scale values and spectral radius values around unity. In the case where p is equal to 10 and d is equal to 14 and 15, the ESN has a higher input scale and in the case where d is equal to 14, a tuned spectral radius value greater than unity. The values of the tuned parameters of the ESN in these last two

cases are likely to have a detrimental impact on its MC where, as the values of d are at their largest, MC is in greater demand. The fact that the R^{2}SP does not have the same high values of input scale and spectral radius where p and d are at their highest could be because the non-linearity required from the dataset is performed mainly by $stat_1$ allowing its reservoir to be tuned towards the maximum MC possible as indicated by its parameters. As with the previous tasks, the MC and deviation from linearity of both architectures when applied to this dataset were calculated and are plotted in Figures 6.12 and 6.13 respectively.

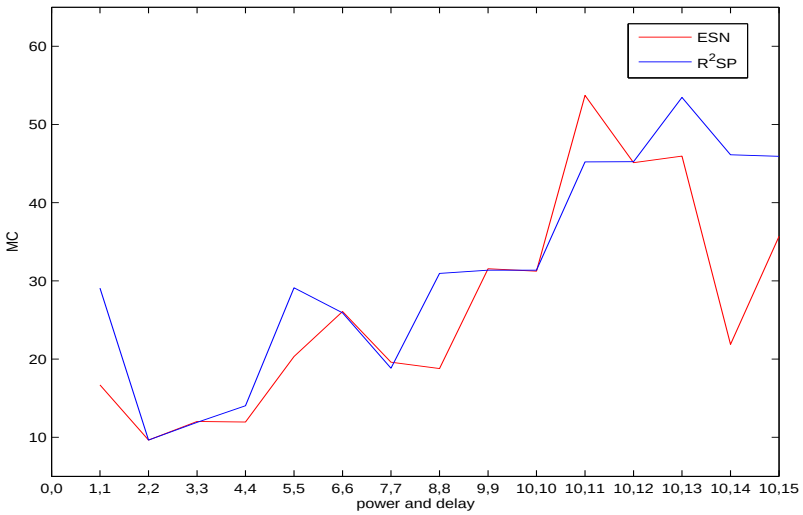


Figure 6.12: The memory capacity of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the power and delay for the extended XOR task. After p reaches a value of 10, it stays equal to 10 while the value of d increases. The MCs of both architectures are similar, with the R^{2}SP having slightly higher values in 8 out of the 15 experiments, which could be the reason for a significant improvement in performance. In the cases where p and d have values equal to 5, the increased MC of the R^{2}SP does not aid in performance as the ESN significantly outperforms it in this case.

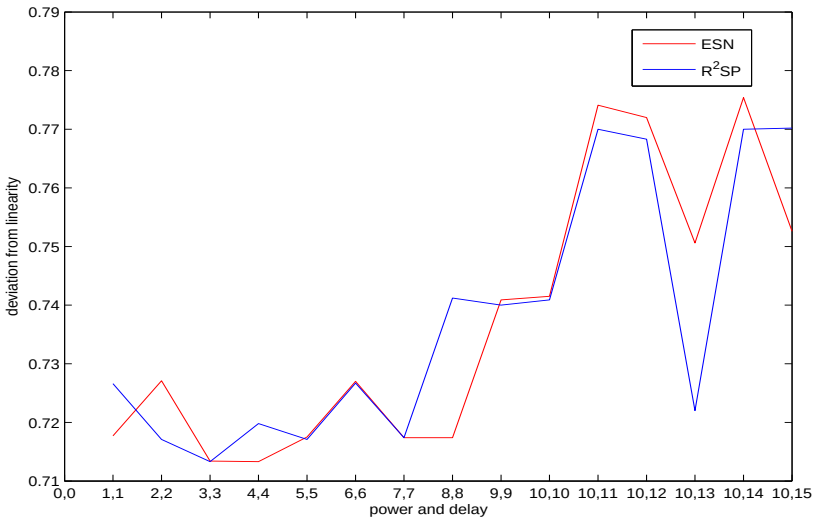


Figure 6.13: The deviation from linearity of an ESN (red) and the R^{2}SP-reservoir (blue) when increasing the power and delay for the extended XOR task showing similar values for each increment of p and d. After p reaches a value of 10, it stays equal to 10 while the value of d increases. Overall, the ESN has a slightly higher deviation from linearity in 9 out of the 15 experiments, although the values for both never become large.

Analysis of Figure 6.12 shows that the MC of the two architectures is similar, with the R^{2}SP having a slightly higher MC in 8 out of the 15 experiments, with the exception of where p is equal to 10 and d is equal to 11 which is caused by a slightly larger tuned input scale of the R^{2}SP (0.63 compared to 0.25 of the ESN). As expected as a result of its tuned parameters (smaller input scale and spectral radius of unity) when p is equal to 10 and d is equal to 14 and 15, the MC of the R^{2}SP is higher than the ESN which may explain its significant improvement in performance: its R^{2}SP-reservoir layer is tuned more towards maximum MC since less non-linear separation is required from it when compared to the ESN, which needs to perform both of these requirements of the dataset. Overall the MC of both architectures tends to increase with increasing p and

d values. This was also shown by Verstraeten et al [226] for an ESN. Since the ESN is also required to perform some non-linear mapping of the input, its MC is slightly less in most cases as a result.

Figure 6.13 shows that, as with their MCs, the two architectures have a similar deviation from linearity over all increments of p and d. Neither architecture has an overly high score, as was also the case for the extended polynomial task (seen in Figure 6.4 previously). Figure 6.13 generally shows that the values of deviation from linearity increase when p is equal to 10 and d is greater than 10, as a result of the tuned spectral radius of both architectures having a value around unity. The ESN generally has a slightly higher deviation from linearity, with a higher score observable in 11 out of the 15 different experiments.

6.3.4 Spoken digit recognition task

6.3.4.1 Training and testing with noisy utterances

Initially, the performance of an ESN and an R^{2}SP were compared when applied to this dataset by training both architectures on spoken digits with added noise using an Signal-to-Noise Ratio (SNR) of 3 dB. An SNR of 3 dB was chosen as this introduces a significant amount of noise into the spoken digits making the task relatively difficult, without introducing too much noise into the spoken digits such that the network would try to capture the characteristics of the noise rather than the spoken utterances. When applied to this task, the ESN's size was set at 150 neurons, while each the R^{2}SP-reservoir, $stat_1$ and $stat_2$ layers of the R^{2}SP also contained 150 neurons. Varying the leak rate for this task was found to affect performance as the data required a network to possess more memory than other datasets investigated here. This is due to the fact

that speech utterances have a larger temporal extent; therefore, in order to remember previous samples and offer good classification performance, the reservoir's short-term memory must stretch further into the past.

As both architectures were trained using noisy data, further regularisation through cross-validation with a grid search and ridge regression was not used as the noise from the data would regularise both architectures sufficiently. Both architectures were also trained using the pseudo-inverse algorithm outlined in Chapter 3. All 500 spoken digits were used during cross-validation: the training set consisted of 450 spoken digits, while the test set consisted of 50 spoken digits, which were shuffled 10 times such that every utterance was used once for testing. Table 6.7 shows the errors, standard deviations, tuned parameters and computation times for both architectures. Figure 6.14 shows the error plots of the best performing ESN (left) and the R^2SP approach (right) when ranging over the leak rate and spectral radius. For this spoken digit recognition task and all of the variations of the training and testing data used here and in the next section, an input scale of 2.5 was used for $stat_1$ and $stat_2$.

Table 6.7: Average test Word Error Rates (WERs) (with standard deviations shown in parentheses), tuned values of the input scale (ι), spectral radius (ρ) and leak rate parameters and computation times obtained from an ESN and an R^{2}SP network that were trained on 450 noisy utterances of spoken digits and tested on the remaining 50 noisy utterances corrupted with an SNR of 3dB using 10-fold cross-validation to train the output weights. The best score is shown in bold.

Type	**WER (std-dev)**	ι	ρ	**Leak rate**	**Computation time** (hrs:mins:secs)
ESN	0.1854 (0.0198)	1	1.3	0.1	00:01:58
R^{2}SP	**0.1459 (0.0159)**	1	1.1	0.1	00:02:46

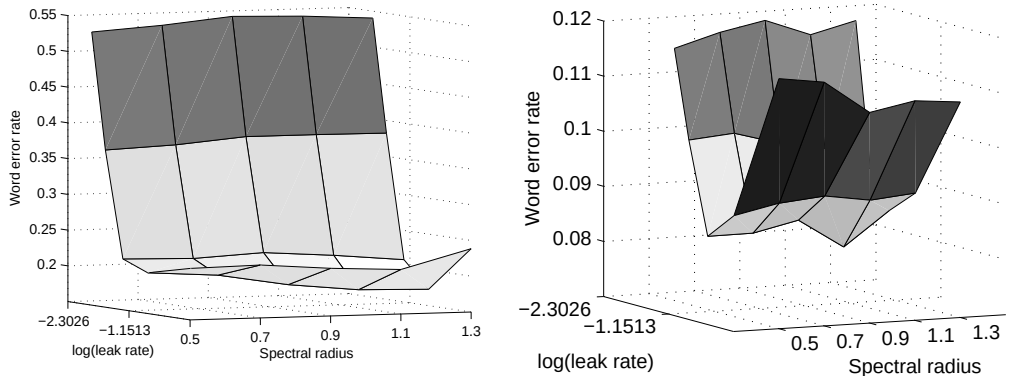


Figure 6.14: Test error plots of the best ESN (left) and the best R^{2}SP (right) when applied to the digit recognition task ranging over the leak rate and the spectral radius. Note the different scale on the Z axis (WER).

As Table 6.7 shows, the R^{2}SP offers superior performance when compared to the ESN. Using the Mann-Whitney U-test for significance reveals that the two errors are

significantly different (p value of $4.42 \times 10^{-27} < 0.05$). Analysis of the tuned parameters reveals little difference between the two architectures, with a higher spectral radius of the ESN the only difference. The table also shows an increase of roughly 50 seconds in computation time for the R^{2}SP which was caused by an increase in the number of output weights to calculate. Given the significant improvement in performance however, this is not problematic. Figure 6.14 shows that the best R^{2}SP offers a WER of 0.08, whereas the best ESN is only able to achieve a WER of 0.17. The figure also shows that when ranging over the spectral radius the value which enabled best performance of the R^{2}SP is lower than that of the ESN. To gain further insight into the properties of both architectures' reservoirs based on the parameters which enabled best performance, their maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), MC and ASE were calculated and are shown in Table 6.8.

Table 6.8: The average maximum local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), memory capacity (MC) and average state entropy (ASE) of the best performing ESN and an R^{2}SP networks for the digit recognition task when trained on 450 noisy samples (standard deviations shown in parentheses).

Type	$\widetilde{LLE}_{max}$	$\delta_{\phi_{res}}$	$\delta_{\phi_{stat_1}}$	$\delta_{\phi_{stat_2}}$	$\delta_{\phi_{comb}}$	**MC**	**ASE**
ESN	0.12	0.8216	n/a	n/a	n/a	10.73	0.03
	(0.0175)	(0.1474)				(2.14)	(0.106)
R^{2}SP	0.08	0.8074	0.7079	0.8121	0.7766	16.87	-1.31
	(0.0156)	(0.0727)	(3.48×10^{-6})	(0.0743)	(0.0652)	(1.78)	(0.2229)

Inspection of Table 6.8 shows that both architectures have an $\widetilde{LLE}_{max}$ value greater than zero, which may be an indication of instability within both networks. The ESN's

$\widetilde{LLE}_{max}$ value is higher than the R^{2}SP's, indicating that if both networks are unstable, the ESN may be more so. The likely cause of this is the spectral radii values which are greater than unity for both architectures. As expected, a higher spectral radius which enabled best performance of the ESN has reduced its MC when compared to the MC of R^{2}SP. This higher spectral radius has also increased the ESN's deviation from linearity. As was the case with the fourth order polynomial, the ESN also has a higher ASE as a result of it transforming the input data onto a higher dimensional state space. In the case of the R^{2}SP-reservoir, it maps its inputs onto a lower dimensional state space as the majority of the non-linear mapping is performed by the static feedforward layers giving it a lower ASE value as a result of its parameters which enabled best performance.

The performance and analysis of the dynamics of the R^{2}SP when trained and tested on noisy spoken digits confirms again that by combining a reservoir with feedforward memoryless layers its R^{2}SP-reservoir layer is tuned towards maximising its MC as it is required to perform little non-linear mapping of the input, which is performed by its additional layers. In this task, where the input is highly non-linear and requires memory of previous inputs, tuning these two requirements separately once again helps to overcome the antagonistic trade-off present within an ESN, giving improved performance.

6.3.4.2 Training on clean utterance, testing on clean and noisy utterances

The performance of the ESN and R^{2}SP architectures was further investigated using the same digit recognition dataset, but using training data consisting of 300 random utterances of clean spoken digits and test data consisting of the remaining 200 unseen utterances which were both clean and corrupted with added noise of various SNRs

ranging from 0.5 to 30 dB (note: each utterance was allowed to be randomly chosen only once). As noise was not present in the training data, 10-fold cross-validation using a grid-search for the optimal regularisation parameter was used during training with ridge regression to calculate the output weights. A TD-ELM's performance was also investigated for this task where, after some searching, a tuned window size of 30 time steps was found to give best performance when the ELM contained 450 neurons. Table 6.9 shows the errors obtained from an ESN, an R^2SP and a TD-ELM when trained with clean spoken digits and tested on the remaining 200 randomly selected utterances corrupted by various levels of noise, while Table 6.10 shows their parameters which enabled best performance, along with their computation times required for training.

Table 6.9: Average test Word Error Rates (WERs) obtained from an ESN, the R^{2}SP and TD-ELM on the spoken digit recognition dataset with various levels of added noise (in dB) with the best scores highlighted in bold (standard deviations shown in parentheses).

Noise	**WER (std-dev)**		
SNR	**ESN**	**R^{2}SP**	**TD-ELM**
0.5	0.359(0.0435)	0.3094(0.0286)	**0.2625(0.0241)**
2	0.336(0.0396)	0.3009(0.0327)	**0.2601(0.0246)**
3	0.3209(0.0436)	0.2778(0.0341)	**0.2394(0.0301)**
5	0.274(0.0401)	0.2411(0.0304)	**0.1823(0.0246)**
8	0.2358(0.029)	0.2049(0.0241)	**0.1707(0.0182)**
10	0.2196(0.0365)	0.194(0.0248)	**0.1426(0.068)**
20	0.1322(0.0355)	0.0858(0.0303)	**0.0506(0.0175)**
30	0.0722(0.025)	0.0301(0.0127)	**0.0226(0.0087)**
Clean	0.0407(0.0132)	0.023(0.0124)	**0.0179(0.0087)**

Table 6.10: Tuned input scale (ι), spectral radius (ρ) and leak rate (δ) parameters and computation times for an ESN, R^{2}SP and TD-ELM when trained on 300 clean random spoken digits. Under each architecture are the topological settings of each network in the form $hl(nn)$ where hl is the number of hidden layers and nn is the number of neurons contained within each hidden layer.

Type	ι	ρ	δ	**Computation time** (hrs:mins:secs)
ESN 1(150)	1	1.4	0.3162	00:04:44
R^{2}SP 3(150)	1	1	0.03	00:20:12
TD-ELM 1(450)	100	n/a	n/a	00:21:07

Table 6.9 shows that when training all three architectures with clean spoken digits and testing both with various noisy unseen digits the TD-ELM delivers the best performance for over all SNRs when compared to the performance of the ESN and R^{2}SP. This improved performance of the TD-ELM was proved as significant using the Mann-Whitney U-test when compared to the ESN and R^{2}SP over all SNRs with `p` values below 0.05. Table 6.9 also shows that the R^{2}SP approach outperforms the ESN over each variation of SNR. Using the Mann-Whitney U-test for significance revealed that the two errors for each SNR are significantly different with `p` values below 0.05.

Inspection of Table 6.10 shows that the ESN has a much higher spectral radius than the R^{2}SP-reservoir as well as a higher leak rate, while the two architectures have the same tuned input scale. As a result of an increased number of neurons and, hence, output weights to calculate using cross-validation with the more intensive grid search, the training time of the R^{2}SP is roughly five times longer than the ESN, giving a trade-off between performance and training time. The table also shows a high input scale for the TD-ELM, which would shift its neurons more towards their non-linear regions. As an optimal window size of 30 time steps was found for the TD-ELM, the windowing of the data into 30 time steps which were presented at each time step increased the size of the input layer (recall that the original dataset contained 77 inputs at each time step, using a window of 30 meant that 2310 input neurons were required). This larger input layer of the TD-ELM explain its longer training time when compared to the other two architectures as shown in Table 6.10. Given the significant improvement in performance, the increase in training times of the R^{2}SP and TD-ELM approaches (which are still much faster when compared to the traditional RANN training algorithms presented below in Table 6.12) are not problematic. As with the other datasets, the best performing networks of an ESN and R^{2}SP were analysed by calculating their deviation from linearity, $\widetilde{LLE}_{max}$, MC and ASE which are shown in Table 6.11 below.

Table 6.11: The average maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), memory capacity (MC) and average state entropy (ASE) of an ESN and an R^{2}SP for the digit recognition task when trained on 300 random clean utterances.

Type	$\widetilde{LLE}_{max}$	$\delta_{\phi_{res}}$	$\delta_{\phi_{stat_1}}$	$\delta_{\phi_{stat_2}}$	$\delta_{\phi_{comb}}$	**MC**	**ASE**
ESN	0.12	0.80	n/a	n/a	n/a	8.42	0.38
	(0.0334)	(0.1758)				(1.46)	(0.0242)
R^{2}SP	-0.002	0.77	0.71	0.77	0.72	10.87	-3.78
	(0.0035)	(0.011)	(3.6396×10^{-6})	(0.0126)	(0.0017)	(1.31)	(0.1948)

Table 6.11 indicates a potentially unstable ESN as its $\widetilde{LLE}_{max}$ is higher than zero caused by its highly tuned spectral radius. This high spectral radius has also caused the ESN's reservoir neurons to move towards their non-linear regimes as indicated by a higher deviation from linearity, which is reflected in a lower MC when compared to the best performing R^{2}SP. Once again, the ESN has a higher ASE value than the R^{2}SP-reservoir as a result of the R^{2}SP-reservoir being tuned towards maximum MC rather than non-linear mapping as well as MC of the input data. For this spoken digit recognition task it is clear that, by separating non-linearity and input recall using the memoryless feedforward layers and a reservoir respectively in the R^{2}SP approach, the antagonistic trade-off often present in an ESN is better addressed once more. This is further confirmed with the use of the TD-ELM, which adopts a similar approach to R^{2}SP, since this significantly outperforms both architectures.

As a further comparison, EN, LRN and DTDNN RANNs were trained using the same 300 random clean samples for training and the remaining 200 samples for testing

over various SNR levels. All of the RANNs were trained for 300 epochs or until the network's validation error increased six consecutive times. Using the training algorithm with the adaptive learning rate a momentum rate of 0.9 was used. Table 6.12 shows the best network parameters as well as their topology, errors and training computation times.

Table 6.12: Average test Word Error Rates (WERs, standard deviation in parentheses) obtained from an ESN, R^{2}SP, TD-ELM, Elman Network (EN), Layer Recurrent Network (LRN) and a Distributed Time-Delay Neural Network (DTDNN) on the spoken digit dataset with the lowest score for each SNR value shown in bold along with the computation times (given in hrs:mins:secs) for training. Topological settings (i.e. the number of hidden layers and neurons in each layer) for each network are shown in the form *hl(nns)* where *hl* is the number of hidden layers and *nns* is the number of neurons in each layer.

Type	**WER (std-dev)**				**Computation**
	Clean	**30 dB**	**20 dB**	**10 dB**	**Time**
ESN	0.0407	0.0722	0.1322	0.2196	00:04:44
1(150)	(0.0132)	(0.025)	(0.0355)	(0.0365)	
R^{2}SP	0.023	0.0301	0.0857	0.194	00:20:12
3(150)	(0.0124)	(0.025)	(0.0355)	(0.0365)	
TD-ELM	**0.0179**	**0.0226**	**0.0506**	**0.1426**	00:21:07
1(450)	**(0.0087)**	**(0.0087)**	**(0.0175)**	**(0.0368)**	
EN	0.204	0.21	0.261	0.36	285:18:58
1(30)	(0.0472)	(0.0557)	(0.0649)	(0.0374)	
LRN	0.179	0.186	0.231	0.336	254:39:41
1(30)	(0.0537)	(0.0726)	(0.092)	(0.0625)	
DTDNN	0.89	0.895	0.92	0.925	31:51:08
1(10)	(0)	(0)	(0)	(0)	

Table 6.12 shows that none of the traditional RANNs outperformed the ESN, R^{2}SP or TD-ELM over the various levels of noise for the digit recognition task. Out of the

three traditional architectures, the LRN offered the best performance, although this was much worse than the RC and ELM approaches. The improvement in performance of the RC and ELM architectures was confirmed as significant using the Mann-Whitney U-test for all test datasets. The table also shows the training times of each architecture, where by comparison the advantages of RC training can be clearly seen. All five DTDNN networks stopped training after only 62 iterations, which explains their lower training times and higher error rates. In comparison, the EN trained on average for 151 epochs (with a standard deviation of 46.81), while the LRN trained on average for 175 epochs (with a standard deviation of 61.14) which explains their longer training times and improved error rates. The fact that the DTDNN networks stopped training after such a small number of iterations could be due to poor convergence caused by local minima. Whilst the EN and LRN offered a significant improvement in performance compared to the DTDNN, their best error score is still far poorer than the scores of the RC and ELM architectures which is also probably due to the problems typically associated with traditional gradient descent RANN training algorithms as were discussed in Chapter 3 previously.

As the R^2SP and TD-ELM architectures have been shown to offer superior performance when compared to an ESN, these results have also been compared to other studies that have applied various machine learning techniques to the same TI46 spoken digit dataset. One such study by Verstraeten et al [247] applied an LSM to the same dataset using 300 random clean spoken samples for training and 200 random samples for testing. Verstraeten et al [247] showed that the LSM could deliver better test performance than other techniques such as an Hidden Markov Model (HMM) when using clean samples as training data and noisy samples corrupted with SNR ratios of 30, 20 and 10 dB as testing data. Other techniques have also been applied to this dataset

using 300 random samples for training and the remaining samples for testing: these are also presented below for further comparison to the TI46 results reported above. A Long Short-Term Memory (LSTM) spiking ANN [335, 336] was applied to this dataset by Graves et al [337] where a test WER of 0.02 was achieved when tested on 200 random clean spoken digits. In Graves et al's study [337], the data was preprocessed using a mel-frequency cepstrum coefficient (MFCC) analysis which computes cepstral coefficients and their approximated first and second order time-derivatives, rather than pre-processing using the Lyon Cochlea model used here (as described earlier in Chapter 5).

A Temporal Reservoir Machine's (TRM) performance was investigated using the same dataset when tested on 200 random utterances of spoken digits corrupted with noise using 5 dB SNR [249]. The TRM was also compared to a Conditional rating Restricted Boltzmann Machine (CRBMs) where it was shown to offer superior performance. A further study by Hermans and Schrauwen [251] also applied a hybrid approach to the same dataset where the input weights and output weights were trained to give a One Step Backpropagation (OSBP) ESN. The performance of a reservoir trained using the OSBP approach was compared with an ESN trained using Recursive Least Squares (RLS-ESN), after both networks were trained on a random set of 250 spoken digits and tested on the remaining 250 clean spoken digits. While this set-up is slightly different from the one used in this research, it is mentioned for completeness. Table 6.13 shows the performance of these architectures presented in other studies. For comparison, the performances of an ESN, R^2SP and TD-ELM from Table 6.9 are also shown.

Table 6.13: Test Word Error Rates (WER) reported from an LSM [247], a TRM, CRBM and HMM [249], an OSBP trained ESN (OSBP-ESN) and an ESN trained with RLS (RLS-ESN) [251] and an LSTM [337] compared against an ESN, an R^2SP and a TD-ELM over various different levels of noise with the best scores highlighted in bold.

Type		**SNR (dB)**			
	Clean	**30**	**20**	**10**	**5**
ESN	0.0407	0.0722	0.1332	0.2196	0.274
R^2SP	0.023	0.0301	0.0857	0.194	0.2358
TD-ELM	0.0179	**0.0226**	**0.0506**	0.1426	0.1785
LSM	0.025	0.055	0.065	**0.11**	n/a
TRM	n/a	n/a	n/a	n/a	**0.07**
CRBM	n/a	n/a	n/a	n/a	0.387
HMM	n/a	n/a	n/a	n/a	0.366
OSBP-ESN	**0.0015**	n/a	n/a	n/a	n/a
RLS-ESN	0.03	n/a	n/a	n/a	n/a
LSTM	0.02	n/a	n/a	n/a	n/a

Analysis of Table 6.13 shows that the OSBP-ESN approach offers the smallest test WER when applied to clean test digits, while the TD-ELM gives the closest WER in comparison. The R^2SP offers comparable performance against an LSTM, RLS-ESN and LSM, all of which outperform the standard ESN approach. The superiority of the OSBP-ESN approach is probably due to the additional training of the input weights, which deviates from the core idea of RC: where only the output weights are usually trained. This improved OSBP-ESN performance is obtained at the expense of slower

convergence compared to standard RC training. The table also shows that the TD-ELM approach outperforms an LSM, R^{2}SP and an ESN when classifying data which has been corrupted with noise over SNRs of 30 and 20 dB, while the LSM offers the best performance for an SNR of 10 dB. The TRM approach, which required a total of 1,200 neurons, offers by far the best classification for an SNR of 5 dB, while the R^{2}SP, ESN and TD-ELM offer superior performance compared to the CRBM and state-of-the-art HMM approaches at this SNR level.

A final comparison between the R^{2}SP, TD-ELM and HMMs is discussed here where they have once again been applied to the TI46 dataset. The performance of an HMM based system, Sphinx4 [338], was investigated by Walker et al [339] on numerous speech recognition tasks, with a WER of 0.00168 reported for the TI46 dataset when using clean test data. As with the LSTM network discussed above, Walker preprocessed the data using MFCC analysis. The error of the ESN, R^{2}SP and TD-ELM are generally worse than the Sphinx4 architecture. However, it could be argued that all three architectures are more robust against noise when compared to the HMM and CRBM reported in Schrauwen and Büsing [249] when using speech samples with a low SNR (5 dB) where the reservoir-based approaches offered superior performance. ANNs' robustness to noise when compared to an HMM has also been reported by Verstraeten et al [247] and Graves [337]. HMMs are also hard to use to perform additional tasks on the same data (for example, speaker identification [340]) without a large increase in computational complexity [210].

6.3.5 Laboratory controlled mesh data

Once all the data had been collected over a 57 week period using the scanning and energising EMAD procedure outlined previously in Chapter 5, the dataset consisted of a total of 1280 scan lines, each of which contained 752 samples, giving a total of 962,560 datapoints. This data was normalised between the range 0 and 1 and sampled by extracting every 10^{th} data point from the normalised data to reduce the size of the dataset. The spatial resolution of the EMAD technique meant that a sampling interval of 10 could be used as the characteristics of the defect signatures were not lost, despite the temporal resolution reduction. Once labelled, the data were randomly split into 680 scan lines for training and 600 for testing. In total, 22.95% of the training data contained data points labelled as defects (i.e. a target value of +1), while 25.82% of the test data contained data points labelled as defects. The remaining data was labelled as defect-free (i.e. a target value of -1).

As the target data contains only two class labels, 'defect' and 'no defect' all of the networks' error measures on the two concrete datasets (the laboratory controlled mesh data presented in this section and the Temple Moor data presented in the next section) were calculated using the exclusive loss error measure (E_{ex_loss}) [227]. This shows the number of incorrectly classified instances and is calculated using the number of datapoints correctly identified as a correct true positive (TP, i.e. a 'defect' is classified as a 'defect'), the number of datapoints incorrectly identified as a false positive (FP, i.e. a 'no defect' is classified as a 'defect'), the number of datapoints correctly identified as true negatives (TN, i.e. a 'no defect' is classified as a 'no defect') and the number of datapoints incorrectly identified as false negatives (FN, i.e. a 'defect' is classified as a 'no defect'). The E_{ex_loss} is defined by:

$$E_{ex\_loss} = \frac{FP + FN}{TP + FP + TN + FN} \tag{6.1}$$

where

$$FP = \sum_{t=1}^{T} (\hat{\mathbf{y}}(t) \geq \Theta \text{ AND } \mathbf{y}_{tgt}(t) = -1) \tag{6.2}$$

$$FN = \sum_{t=1}^{T} (\hat{\mathbf{y}}(t) < \Theta \text{ AND } \mathbf{y}_{tgt}(t) = +1) \tag{6.3}$$

$$TP = \sum_{t=1}^{T} (\hat{\mathbf{y}}(t) \geq \Theta \text{ AND } \mathbf{y}_{tgt}(t) = +1) \tag{6.4}$$

$$TN = \sum_{t=1}^{T} (\hat{\mathbf{y}}(t) < \Theta \text{ AND } \mathbf{y}_{tgt}(t) = -1) \tag{6.5}$$

where T is the total number of time steps in the dataset, $\hat{\mathbf{y}}$ is the actual output of the network, $\mathbf{y}_{tgt}$ is the target output and Θ is a threshold (throughout this research Θ was set at 0). While this measure gives a good indication of the performance of the network, it can sometimes be misleading. For example, if a dataset contains 80% of datapoints labelled as 'no defect' and 20% labelled as 'defect', then an error of 0.2 using the exclusive loss error measure would initially indicate a network that offers reasonable performance. However, this may not be the case as it may have learned to classify everything as 'no defect' and have therefore learned nothing. To overcome this problem and give more insight into the performance of the networks, the values of TP, FP, TN and FN were used to calculate their sensitivity and specificity according to equations 6.6 and 6.7.

$$Sensitivity = \frac{TP}{TP + FN} \tag{6.6}$$

$$Specificity = \frac{TN}{TN + FP} \tag{6.7}$$

where the sensitivity shows the proportion of 'defects' successfully detected by the network as such and the specificity shows the proportion of 'no defects' successfully detected by the network as such.

Further analysis was performed by calculating the positive predictive value (PPV) which is the proportion of the ground truth datapoints labelled as 'defects' out of all the datapoints classified as 'defects' by the network and the negative predictive value (NPV) which is the proportion of the ground truth 'no defect' datapoints classified by the network as 'no defects'. These two measures are defined in equations 6.8 and 6.9.

$$PPV = \frac{TP}{TP + FP} \tag{6.8}$$

$$NPV = \frac{TN}{TN + FN} \tag{6.9}$$

As a final complementary performance measure, the Matthews correlation coefficient (MCC) [341, 342] was calculated as follows:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}} \tag{6.10}$$

An MCC score of unity indicates perfect classification, whereas a score of 0 indicates random guessing. The MCC was used to complement the previous measures and to indicate how well the network was performing, where a low MCC value would indicate poor performance which may have been undetectable using some of the other measures.

When applied to this dataset, the size of the reservoir of the ESN and the layers of the R^{2}SP were set at 200 neurons. Varying the bias constant, β, was also found to make little difference in performance and this was kept at the default value of zero. In addition to these parameters, varying the input scale of $stat_1$ was found to improve performance, where a large value of 100 was found to offer the highest classification rate. The window size of the data presented to the TD-ELM was ranged over and a size of 15 datapoints was found to give best performance (note that the size of the window for the EMAD datasets was not allowed to exceed 30 data points where larger window values may artificially improve the TD-ELM's performance). Table 6.14 shows errors and the values of the input scale, spectral radius and leak rate for the ESN, R^{2}SP and TD-ELM architectures which enabled best performance. Further analysis of the performance of all three architectures by calculating the sensitivity, specificity, PPV, NPV and MCC was conducted as shown in Table 6.15 below.

Table 6.14: The average exclusive loss error rate (E_{ex_loss}) (standard deviations shown in parentheses), tuned input scale (ι), spectral radius (ρ), and leak rate parameters and computation times of an ESN, the R^2SP and the TD-ELM when applied to the laboratory controlled mesh data. The best classification error is shown in bold. Topological settings (i.e. the number of hidden layers and neurons in each layer) for each network are shown in the form *hl(nns)* where *hl* is the number of hidden layers and *nns* is the number of neurons in each layer.

Type	E_{ex_loss} **(std-dev)**	ι	ρ	**leak rate**	Computation Time
ESN 1(200)	0.1603(0.0044)	0.25	0.9	0.3981	00:13:33
R^2SP 3(200)	**0.1356(0.0028)**	1	0.9	0.3981	00:46:36
TD-ELM 1(600)	0.1825(0.0024)	100	n/a	n/a	01:19:43

Table 6.15: The sensitivity (Sens), specificity (Spec), PPV, NPV and MCC of an ESN, R^{2}SP and TD-ELM (standard deviations shown in parentheses) when applied to the laboratory controlled mesh data with the best scores shown in bold.

Type	**Sens**	**Spec**	**PPV**	**NPV**	**MCC**
ESN	**0.9262**	0.80	0.6289	**0.9693**	0.6634
	(0.0052)	(0.0067)	(0.0076)	**(0.002)**	(0.0069)
R^{2}SP	0.9219	**0.8444**	**0.6735**	0.9688	**0.7016**
	(0.004)	**(0.0043)**	**(0.0056)**	(0.0015)	**(0.0045)**
TD-ELM	0.8701	0.7992	0.6013	0.9465	0.6055
	(0.0053)	(0.004)	(0.0041)	(0.0019)	(0.0035)

As Table 6.14 shows the R^{2}SP architecture offers the smallest average error rate. A ESN's best error rate was 0.15, while the best R^{2}SP gave an error of 0.1295 compared to a TD-ELM's best score of 0.1766. The R^{2}SP significantly outperformed the TD-ELM architecture with a p value of 5.44×10^{-34} (< 0.05) calculated from the Mann-Whitney U-test. A p value of 7.95×10^{-34} (< 0.05) was calculated when comparing the R^{2}SP and the ESN making the the R^{2}SP's improvement in performance further significant. The difference between the performance of the ESN and the TD-ELM was found to be significant using a 95% confidence level Mann-Whitney U-test (a p value of 7.95×10^{-34} was calculated). The training times of the three architectures are also given. The training times for each architecture are much slower than those previously reported from the other case studies as a result of the large training dataset. From these training times it is clear that the ESN offers the fastest training, while the R^{2}SP is roughly four times slower. The TD-ELM has by far the slowest training time due to

the large input layer required to accommodate the window size of 15 datapoints (as the input data requires 3 input neurons and with a window of 15 datapoints, 45 neurons are required).

Table 6.14 also shows that the parameters which gave best performance for the ESN and R^2SP architectures are very similar, where both the spectral radius and leak rate have the same value. Only the input scale to the reservoir of the two architectures differs, with the value for the ESN smaller than unity, which may increase its MC. Analysis of the sensitivity and specificity of the ESN, R^2SP and TD-ELM architectures (shown in Table 6.15) reveals that the ESN has a slightly higher sensitivity than the R^2SP but a lower specificity as the R^2SP gives the fewest false positives which is also desirable when on-site, where removing concrete from a healthy section of a structure is a costly exercise. The superior exclusive loss error rate and MCC of R^2SP in comparison to the ESN are due to its increased specificity. Since more non-defect datapoints are present within the dataset, fewer false positives gives a lower error rate in comparison to fewer false negatives. As the TD-ELM has the highest error rate, its MCC is the lowest of the three architectures. Since the sensitivity of the R^2SP is only slightly less than that of the ESN, its overall performance is superior as a result of fewer false alarms.

The R^2SP also scores a higher PPV than an ESN which means that out of the datapoints labelled as defects, this architecture classifies 'defects' more accurately, while the NPV values for the two reservoir architectures are similar indicating that they correctly classify a similar proportion of the ground truth 'no defect' datapoints from all of the datapoints classified as 'no defect'. The MCC of the three architectures confirms that the R^2SP offers the best performance.

Further analysis of the performance of the ESN, R^2SP and TD-ELM architectures was performed by analysing the datapoints that the networks classified as 'defect' and calculating the percentage of specific defect types detected (defect types were outlined previously in Chapter 5). These scores were calculated by comparing the number of datapoints a network signalled as a defect by the number of labelled 'defect' datapoints in the target data. For example, if in the target data on scan line five, defect 4B was labelled between datapoints 30-40 and a network gave an output of +1 (i.e. a 'defect' was detected) between datapoints 32 and 40 inclusive, then the network will have detected 80% of that particular 4B defect. This is shown in Table 6.16 below.

The 'granularity of the network' analysis approach above uses individual datapoints and, one could argue, is a very strict approach to assess each architecture's capability of detecting defects. At a higher level of granularity if a network gave an output of +1 (i.e. a 'defect' was detected) in a region where a defect was present in the target data, then a successful detection can be recorded. This approach may give higher detection rates but for practical applications where an area of a structure identified as a potential problem would typically involve the removal of concrete around the suspect zone, and is therefore legitimate: any defect present in this region detected by the network would be uncovered by the engineer removing the concrete. Table 6.17 below shows the performance of the three networks when adopting this approach.

Table 6.16: The percentage of the different defects detected by an ESN, R^{2}SP and TD-ELM when applied to the experimental mesh data calculated by comparing the number of defect datapoints indicated by the network against the number of defect datapoints in the labelled target data with the best scores highlighted in bold.

Type	**Percentage of defects detected**							
	4B	5B	7BC	8BC	9B	10C	11BC	12B
ESN	**92.72**	**95.12**	**94.8**	92.25	92.82	**81.17**	94.77	91.54
R^{2}SP	89.22	93.85	93.06	**95.5**	**94.02**	78.49	**95.37**	91.23
TD-ELM	77.54	82.93	92.94	90.67	93.13	76.91	90.63	**93.84**

Table 6.17: The percentage of the different defects detected by an ESN, R^{2}SP and TD-ELM when applied to the experimental mesh data calculated by assessing each architecture's ability to detect a defect within the defective zone with the best scores shown in bold.

Type	**Percentage of defect detected by zone**							
	4B	5B	7BC	8BC	9B	10C	11BC	12B
ESN	**99.62**	**99.9**	**99.86**	99.94	99.37	**95.57**	99.28	**99.16**
R^{2}SP	98.32	99.47	98.86	99.81	99.92	93.05	99.55	98.89
TD-ELM	97.54	99.32	99.57	**100**	**99.95**	93.91	**99.78**	98.85

Table 6.16 shows that the ESN detects a higher percentage of defect datapoints for four of the eight defect types present in the test data. The R^{2}SP detects more datapoints labelled as 8BC, 9B and 11BC, while the TD-ELM detects more datapoints for defect 12B. This was also reflected in Table 6.15 accounting for the marginally higher

sensitivity score of the ESN. As the sensitivity of the ESN and R^{2}SP architectures is close, the percentage differences shown above are also very small, with neither network outperforming the other by more than roughly 3.5%. As the TD-ELM's sensitivity score was lower than the other two approaches, this is also reflected in the majority of defect datapoints it detects.

As Table 6.17 used a less stringent calculation to assess each architecture's ability to detect defects the detection rates for the three architectures are much higher. Again the slight improvement in the ESN's sensitivity compared to the other two architectures is reflected in the higher percentages of defects it detects, detecting more than the R^{2}SP and TD-ELM architectures in five out of eight defect types. As before, the improvement in the ESN's sensitivity is only slight, this time with no difference between the three architectures greater than roughly 2.6%. Using this less stringent calculation appears to favour the TD-ELM which detects the highest amount of the remaining defects, with a 100% detection rate for defect 8BC. While these scores are usually ideal, the TD-ELM's high scores come from its low specificity and PPV scores, which indicate that the TD-ELM is prone to give more false alarms than the other two approaches. As a result of this it classifies more datapoints as defects, hence the higher scores in Table 6.17 above.

To gain insight into the dynamics of the ESN and R^{2}SP architectures the estimated maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), MC and ASE were calculated as shown in Table 6.18 below.

Table 6.18: The maximum local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), memory capacity (MC) and average state entropy (ASE) of an ESN and an R^{2}SP using their parameter settings which enabled best performance for the experimental mesh dataset (standard deviations shown in parentheses).

Type	$\widetilde{LLE}_{max}$	$\delta_{\phi_{res}}$	$\delta_{\phi_{stat_1}}$	$\delta_{\phi_{stat_2}}$	$\delta_{\phi_{comb}}$	**MC**	**ASE**
ESN	-0.1054	0.7482	n/a	n/a	n/a	24.71	-3.77
	(5.48×10^{-6})	(0.002)				(1.1789)	(0.0407)
R^{2}SP	-0.1095	0.7421	0.7745	0.7459	0.7731	20.52	-4.55
	(0.0052)	(0.0027)	(0.0023)	(0.0033)	(0.0022)	(0.9693)	(0.0554)

As the parameters of the two architectures which enabled best performance were very similar, this is reflected in many of their reservoir's dynamics. The ESN and R^{2}SP architectures have a similar $\widetilde{LLE}_{max}$ value less than zero which indicates stability. Likewise, the two architectures have reservoirs with a similar deviation from linearity. The overall deviation from linearity of the whole R^{2}SP network is much higher than that of the ESN as a result of the high input scale to $stat_1$. The smaller input scale of the ESN results in a larger MC than the R^{2}SP as the reservoir is less influenced by the current input, whereas the R^{2}SP-reservoir's input scale of unity increases the influence of the current input on the state of the R^{2}SP-reservoir which slightly hinders its ability to recall datapoints from the past. While for some tasks a reduced MC may be undesirable, in this task the defects do not last very long; therefore a long MC may not be required and, as shown by the improved performance of the R^{2}SP in Tables 6.14 and 6.15 above, it does not impact negatively its performance. The ASE values of both architectures are more negative than in previous case studies, where the R^{2}SP-reservoir

again has a smaller value than the ESN. As before, this may be a result of the ESN's reservoir mapping the input data onto a larger state space as it is required to by the task, as the ESN has to perform non-linear transformation of the input data whilst recalling inputs from the past.

As the R^{2}SP-reservoir has a smaller MC, although the difference between the two architectures's MC is small, its improvement in performance, as indicated by a lower exclusive loss error and a higher MCC, is likely to be a result of the instantaneous reaction of $stat_1$ to the fast changing input data as was also shown in the extended polynomial task above. As was shown in Chapter 5, the data collected by the EMAD can be non-linear and fast changing, thus the capability of $stat_1$ to react to this well is expected to offer improved performance.

6.3.6 Temple Moor High School

This dataset consisted of 191 scan lines, 100 of which were used for training with the remaining 91 used for testing. For both subsets of data, scan lines were chosen at random, although each scan line could only be used once. This data was normalised in the range 0 to 1 and sampled by extracting every 10^{th} data point from the normalised data to reduce the size of the dataset as with the laboratory controlled mesh data. No absolute ground truth data was available here so after the target data was labelled by an expert; 43.55% of the training data contained data points which were labelled as 'defects' and 43.83% of the test data contained data points which were labelled as 'defects'. The performance of an ESN, R^{2}SP and TD-ELM were compared using the exclusive loss error measure (E_{ex_loss}), along with each architecture's parameters which enabled best performance as shown in Table 6.19. Table 6.20 shows further analysis of each architecture's performance with respect to its sensitivity, specificity, PPV, NPV

and MCC scores. As with the previous datasets, the input scales to $stat_1$ and $stat_2$ of the R^{2}SP were also ranged over and values of 100 and 12.58 were found to give best performance for each layer respectively. The window size which affects the TD-ELM's memory, was ranged over to find the best size, which for this dataset was five datapoints. The relatively short length of the window is probably caused by the fact that the defect signatures in the data do not last over a long period, and five time steps along with the sampling of the data is enough to capture enough of the defect signatures to allow best performance.

Table 6.19: The average exclusive loss error rate (E_{ex_loss}), tuned input scale (ι), spectral radius (ρ) and leak rate parameters and computation times of an ESN, R^{2}SP and TD-ELM when applied to the Temple Moor dataset (standard deviations are shown in parentheses). The best score is shown in bold. Topological settings (i.e. the number of hidden layers and neurons in each layer) for each network are shown in the form *hl(nns)* where *hl* is the number of hidden layers and *nns* is the number of neurons in each layer.

Type	E_{ex_loss} **(std-dev)**	ι	ρ	**leak rate**	**Computation time**
ESN 1(200)	0.2636(0.127)	31.63	1.6	0.31	00:02:13
R^{2}SP 3(200)	0.2252(0.0041)	3.16	0.7	1	00:11:16
TD-ELM 1(600)	**0.2219(0.0035)**	100	n/a	n/a	00:10:53

Table 6.20: The sensitivity (Sens), specificity (Spec), PPV, NPV and MCC of an ESN, R^{2}SP and TD-ELM (standard deviations shown in parentheses) when applied to the Temple Moor dataset with best scores shown in bold.

Type	**Sens**	**Spec**	**PPV**	**NPV**	**MCC**
ESN	0.6481	0.8013	0.7225	0.7458	0.4609
	(0.0127)	(0.0113)	(0.0107)	(0.0063)	(0.0135)
R^{2}SP	0.6609	**0.8636**	**0.7907**	0.7655	0.5402
	(0.0096)	**(0.0061)**	**(0.0067)**	(0.0047)	(0.0085)
TD-ELM	**0.6771**	0.8564	0.7865	**0.7731**	**0.5467**
	(0.0106)	(0.0073)	(0.0068)	**(0.0048)**	**(0.0071)**

Table 6.19 shows that the TD-ELM outperforms an ESN and R^{2}SP which, using the Mann-Whitney U-test, was confirmed as significant with p values of 2.54×10^{-34} and 1.77×10^{-8} (both of which are less than 0.05) respectively. Table 6.19 also shows the R^{2}SP approach offers increased performance by roughly 4% when compared to an ESN. This improvement in performance was confirmed as significant using the Mann-Whitney U-test where a p value of $2.54\times10^{-34}(< 0.05)$ was calculated. Inspection of the sensitivity and specificity shown above in Table 6.20 reveals that the TD-ELM and R^{2}SP detect more defects, with the TD-ELM giving the highest sensitivity score. These two architectures also give less false positives compared to the ESN, with the R^{2}SP having the highest specificity score, hence fewer false alarms. The probability of detecting defective areas is also highest when using the R^{2}SP as shown by higher value of PPV, while the TD-ELM has a slightly higher NPV. With the largest MCC value, the superior performance of the TD-ELM is confirmed, while the R^{2}SP is a close

second. As was the case when processing the experimental mesh dataset, the ESN offers the fastest training times, followed by the TD-ELM and R^{2}SP. These two architectures take longer to train as a result of more output weights to calculate (600 in the case of the R^{2}SP and TD-ELM). The TD-ELM also had a larger input layer than the other two architectures where, as a result of the time-windowing of the dataset, 15 input neurons were required. Inspection of the tuned parameters of the ESN shows a very high input scale and spectral radius, compared to low values for the R^{2}SP-reservoir, whose effects on the dynamics of each architecture are shown in Table 6.21 below.

Table 6.21: The maximum local Lyapunov exponents ($\widetilde{LLE}_{max}$), deviation from linearity (δ_ϕ), memory capacity (MC) and average state entropy (ASE) of an ESN and an R^{2}SP using their tuned parameter settings for the Temple Moor dataset (standard deviations shown in parentheses).

Type	$\widetilde{LLE}_{max}$	$\delta_{\phi_{res}}$	$\delta_{\phi_{stat_1}}$	$\delta_{\phi_{stat_2}}$	$\delta_{\phi_{comb}}$	**MC**	**ASE**
ESN	0.3408	0.81	n/a	n/a	n/a	5.19	-0.187
	(0.01)	(0.0012)				(0.25)	(0.048)
R^{2}SP	-0.3919	0.736	0.774	0.7766	0.771	18.61	-0.5021
	(0.0047)	(0.0029)	(0.0024)	(0.0033)	(0.0016)	(0.29)	(0.0134)

Table 6.21 shows that the high values of input scale and spectral radius have a drastic impact on the dynamics of the ESN, resulting in an estimated maximal Lyapunov exponent larger than zero which indicates that its reservoir is potentially unstable. The ESN neurons' working points are shifted towards their non-linear regions as a result of the highly tuned input scale and spectral radius, as indicated by its higher deviation from linearity when compared to the R^{2}SP. As a counter to this, a leak rate of 0.31 for

the ESN increases the MC of the individual neurons somewhat but, as shown by its low MC, this effect is minimal. As the R^{2}SP-reservoir is closer to its linear regime, the added memory provided by decreasing the leak rate is not required as shown by a leak rate of unity. This is probably due to the fact that although memory is required for the task, the defects tend not to be present in the EMAD data for long periods and, hence, a long MC is not required as was shown for the previously investigated mesh EMAD dataset. The very low MC of the ESN offers an explanation as to why its performance is worse than the R^{2}SP suggesting it is instead tuned towards non-linear separability of the fast changing input data and, as a result, is potentially unstable. As the data is also fast changing, the improved performance of the R^{2}SP can be attributed, in part, to the $stat_1$ layer. By separating the MC and non-linearity required by the task the R^{2}SP architecture again tunes its R^{2}SP-reservoir layer mainly towards input recall as the majority of the non-linear transformation is performed by the memoryless feedforward layers $stat_1$ and $stat_2$. As the R^{2}SP-reservoir is tuned towards as high MC as possible, its ASE value is much lower than the ESN, which indicates that the diversity of its neurons activations is smaller than the ESN's.

6.4 Pruning of least significant neurons

The effectiveness of pruning the R^{2}SP networks for some of the datasets presented in this work was also investigated, using the pruning procedure outlined earlier in Chapter 4. This process involved using a ranking algorithm, multiresponse sparse regression (MRSR) [290], to rank the R^{2}SP-reservoir, $stat_1$ and $stat_2$ layers' neurons according to their significance to performance. The least significant neurons were then removed from the network and the output weights recalculated accordingly. With each neuron removal the train and test errors of the R^{2}SP were also recalculated. The

pruning procedure was computationally demanding due to the recalculation of the output weights after each neuron removal, so only five networks for each case study dataset were simulated and pruned.

As the size of the R^{2}SP layers and the other architectures was found using tentative empirical searches, they may not have been as extensively tuned as the other parameters. Despite this, as all RC output weights are trained using Ordinary Least Squares, the weights produced for the networks presented previously were optimal given the tuned network parameters which included the size of the architectures used. Pruning was therefore used to find the network size which enabled best performance, although earlier indications from the tentative searches suggested that any difference in the network size would lead to only slight changes in network performance.

The average test error rate for the optimal size of each layer of the R^{2}SP was calculated when applying each network to the unseen 4^{th} order polynomial, spoken digit and laboratory controlled mesh test data which were reported above in sections 6.3.1, 6.3.4.2 and 6.3.5 respectively. The average errors and optimal layer sizes of the R^{2}SP when applied to these various test datasets are shown below in Table 6.22.

Table 6.22: Errors (Error WP) and average optimal layer sizes obtained when using an R^{2}SP with pruning with the unseen test datasets of the polynomial (Poly), clean and noisy samples from the digit recognition (Clean/Noisy speech) and the laboratory controlled reinforced concrete (Mesh) datasets (standard deviations shown in parentheses). The errors obtained for the unpruned networks reported earlier are shown for comparison (Error WOP).

Dataset	**Error WP**	**Error WOP**	**R^{2}SP-res. size**	$stat_1$ **size**	$stat_2$ **size**	**Neurons removed**
Poly	0.517 (0.0012)	0.5241 (9.5164×10^{-4})	87 (9.63)	4 (2.07)	7 (2.51)	502
Clean speech	0.0112 (0.0048)	0.023 (0.0124)	124 (30.64)	98 (35.27)	88 (65.64)	140
Noisy speech (3 dB)	0.224 (0.0102)	0.2778 (0.0341)	136 (19.11)	123 (31.76)	101 (50.41)	90
Mesh	0.1329 (0.0017)	0.1356 (0.0028)	200 (0)	200 (0)	177 (11.7)	23

Table 6.22 shows that the error for all tasks is lower than the four errors of the unpruned networks reported in Tables 6.1, 6.9 and 6.14 using the same unseen test data. This is due to the optimisation of the size of the network via pruning, something which prior to pruning was not performed as exhaustively as for the other parameters. Pruning therefore helps to regularise the network further by removing the neurons which have a detrimental effect on its performance. Pruning the R^{2}SP when applied to the polynomial task reveals that the majority (502) of R^{2}SP-reservoir, $stat_1$ and $stat_2$ neurons are removed with the most removals from the $stat_1$ and $stat_2$ layers (196 and 193 respectively). This suggests that as a result of the dataset requiring little MC,

the R^{2}SP-reservoir layer performs more non-linear mapping (the tuned input scale of 2.5 for the R^{2}SP-reservoir when applied to this task also suggests this); therefore the two feedforward layers' contributions to the non-linear transformation are less, which is reflected in their small number of neurons retained after pruning. Figure 6.15 shows the error of an R^{2}SP after removing each of its neurons according to their significance as defined by the MRSR algorithm.

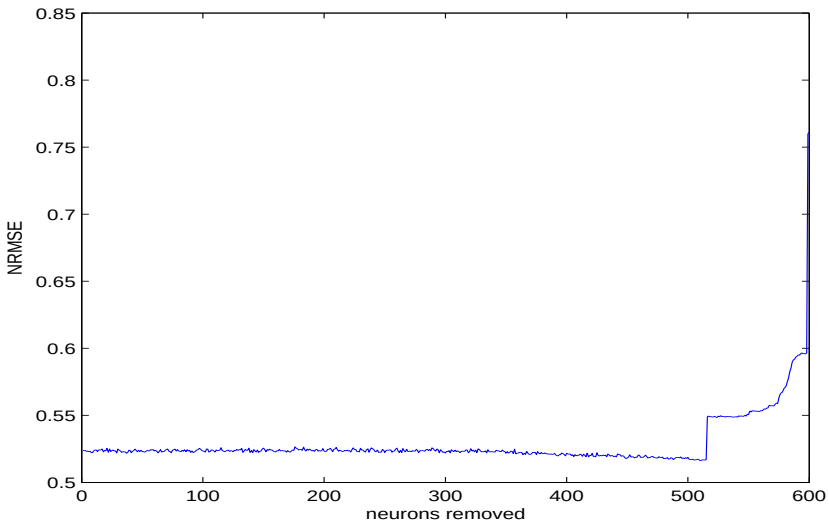


Figure 6.15: The test error of an R^{2}SP after the removal of each neuron based on its contribution to performance when applied to the fourth order polynomial task.

Analysis of the optimal size of the R^{2}SP layer size when applied to the 200 unseen clean spoken digits reveals that the R^{2}SP-reservoir has the highest number of remaining neurons, followed by $stat_1$ and finally $stat_2$. When applied to versions of the same 200 utterances corrupted with an SNR of 3 dB pruning improved the performance of the R^{2}SP more than when applied to the other datasets, with a WER reduction from 0.2778 to 0.224. The total number of neurons removed decreased when compared to the same 200 clean test utterances, as shown by Table 6.22, where 50 fewer neurons were removed.

In this case the R^{2}SP-reservoir still has the highest number of neurons, while the optimal number of $stat_1$ and $stat_2$ neurons had a proportionally higher increase when compared to the R^{2}SP-reservoir from 98 to 123 and 88 to 101 respectively (compared to the increase of the R^{2}SP-reservoir neurons from 124 to 136). The increase in the R^{2}SP layer size when tested with noisy spoken digits may be a result of the increased difficulty of the task due to the reduction in the SNR, which therefore requires more processing power in terms of MC and non-linear separation from each layer to offer superior performance. Figure 6.16 shows the error of an R^{2}SP after the removal of each neuron when processing this dataset.

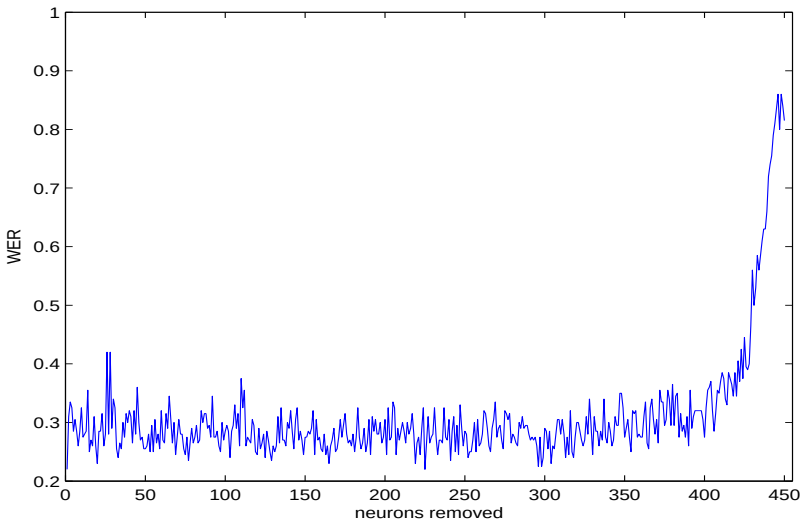


Figure 6.16: The test error of an R^{2}SP after the removal of each neuron based on its contribution to performance when applied to the spoken digit recognition task using 200 samples corrupted with an SNR of 3 dB.

When applying the pruning algorithm to the controlled mesh dataset, Table 6.22 shows a slight decrease in the average error of the R^{2}SP architectures simulated. The table also reveals that none of the R^{2}SP-reservoir and $stat_1$ neurons were removed from

the network, while on average 23 neurons were removed from $stat_2$. This indicates that the dataset required the memory provided by the R^{2}SP-reservoir and the non-linear transformation provided by $stat_1$ more than the polynomial and spoken digit recognition tasks investigated previously. Figure 6.17 shows the error of an R^{2}SP network after the removal of each neuron.

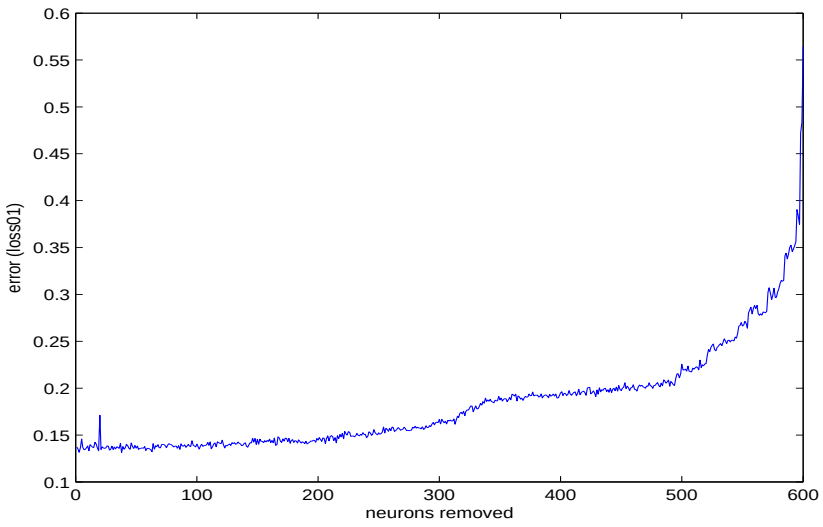


Figure 6.17: The test error of an R^{2}SP after the removal of each neuron based on its contribution to performance when applied to the laboratory controlled mesh dataset.

Analysis of the optimally pruned R^{2}SP networks reveals that pruning helps to improve performance for all three datasets investigated when comparing the test errors using the unseen data. In the cases of the fourth order polynomial, the clean spoken digits and the laboratory controlled mesh datasets this improvement was only slight, with an decrease in error no larger than 1%. Similar reductions in error were also observed by Dutoit et al [314] when pruning an ESN. In the case of the optimally pruned network applied to the noisy spoken digits, the reduction in error is a much higher 5% which is more significant than the reduction in errors of the other datasets. The

decrease in error for each task is a result of the further regularisation provided by the pruning procedure. This additional improvement in performance may indicate that the regularisation performed during training prior to pruning, using ridge regression and cross-validation using a grid search to find the best regularisation for a given set of network parameters, was sub-optimal. However, these approaches to regularise the network will provide an optimal set of output weights based on network parameters, of which the size of the network one. As the size of the network was not tuned as exhaustively prior to pruning (for example, using an empirical search which started with sizes of 50 neurons and increased the size of the reservoir by 50 neurons at a time), further optimisation of the network size down to neuron level using pruning results in the improved performance shown above. Despite these improvements in performance, the pruned R^{2}SP still does not outperform the TD-ELM's performance when applied to the fourth order polynomial and spoken digit recognition datasets.

The pruning conducted in this section has also provided further insight into which R^{2}SP connections are the most useful. For all unseen test datasets, the R^{2}SP-reservoir contained the highest optimal number of neurons. In the case of the polynomial task, very few neurons were required with over 500 of the 600 removed from the entire network (the R^{2}SP-reservoir layer had the highest number of neurons remaining with 87, followed by $stat_2$ with 7 neurons and $stat_1$ with 4 neurons). This may be an indication of the relative ease of the task when compared to the other two tasks. The size of $stat_1$ and $stat_2$ layers for this task is very small, which implies that the R^{2}SP-reservoir performs more of the non-linear transformation while little MC is required for this task. For the spoken digit recognition and laboratory controlled mesh datasets, many more neurons were required which is an indication of the increase in the MC and non-linear mapping requirements for both tasks. Despite the computational intensity

of the pruning approach, the improvements in performance and the added insights provided by pruning have been seen to be worthwhile.

6.5 Summary

This chapter has shown the performance of the ESN, R^2SP and TD-ELM in comparison to traditional RANNs and other state-of-the-art architectures when applied to a variety of tasks. To summarise these results and to indicate which of the hypotheses proposed in Table 4.2 are true, the table below shows these hypotheses proposed earlier in Chapter 4. In the cases where the R^2SP outperforms an architecture for a specific task, this architecture is underlined.

Table 6.23: The different architectures that the R^{2}SP's performance was compared against when applied to various datasets. Here the hypothesis that the R^{2}SP offers a significant improvement in performance when compared to a different architecture is true when the architecture it was compared against is underlined.

Architecture R^{2}SP outperforms	**Dataset**	**Error measure**
ESN, TD-ELM, EN, LRN, DTDNN, EN-BPTT, EN-RTRL	Fourth order polynomial (see Table 6.1)	NRMSE
ESN	Extended polynomial increasing p and d (see Table 6.3)	NRMSE
ESN	Extended polynomial increasing d, $p = 1$ (see Table 6.4)	NRMSE
ESN	Extended polynomial increasing p, $d = 1$ (see Table 6.5)	NRMSE
ESN [2]	Extended delayed XOR (see Table 6.6)	NRMSE

Continued on Next Page...

[2] In 11 of the 15 values of p and d

Table 6.23 – Continued

Architecture R²SP outperforms	**Dataset**	**Error measure**
ESN	Spoken digit task: trained using noisy utterances (see Table 6.7)	WER
ESN, TD-ELM, EN, LRN, DTDNN, LSM, TRM, CRBM, HMM, OSBP-ESN, RLS-ESN, LSTM	Spoken digit task: trained using clean utterances (see Tables 6.9 and 6.13)	WER
ESN, TD-ELM	Laboratory controlled mesh (see Tables 6.14 and 6.15)	% defects incorrect ($E_{ex\llcorner loss}$), Sensitivity, Specificity, PPV, NPV, MCC, % specific defects detected
ESN, TD-ELM	Temple Moor (see Tables 6.19 and 6.20)	% defects incorrect ($E_{ex\llcorner loss}$), Sensitivity, Specificity, PPV, NPV, MCC

As Table 6.23 shows R²SP significantly outperforms an ESN in all of the case studies presented. When applied to the extended delayed XOR dataset, the ESN significantly

outperforms the R^{2}SP in two of the 15 variations over p and d, but the R^{2}SP offers a significant performance in 11 of the 15 variations, therefore offering a significant performance for the majority of values. In two cases (when p and d where equal to 4 and 10), the R^{2}SP gave the best score, but these were not significantly better.

Comparing the R^{2}SP against the conventional RANNs shows that the R^{2}SP offers a significant improvement in performance for the two case studies (fourth order polynomial and spoken digit recognition datasets) where the these types of RANNs were investigated. In fact, the ESN and TD-ELM architectures also significantly outperformed these architectures, showing that the reservoir and ELM-based approach to training neural networks give improved performance as a result of their simplified and efficient training procedures.

As the spoken digit recognition dataset has been used throughout the literature, there were many different architecture's performance to draw comparisons against an ESN, R^{2}SP and TD-ELM. Inspection of Table 6.23 when applied to this dataset using noisy utterances as training and testing data shows that the R^{2}SP outperforms an ESN. As no other studies in the literature had used utterances corrupted with an SNR of 3 dB, these two architectures were the only ones compared using this training and testing set-up. When training using clean spoken utterances and testing using noisy utterances, the R^{2}SP was shown to significantly outperform an ESN and RLS-ESN, the conventionally trained RANNs, as well as the state-of-the-art CRBM and HMM when tested on utterances corrupted with various SNRs. The R^{2}SP gave a significant improvement in performance when compared to an LSM for test utterances corrupted using an SNR of 30 dB, but the LSM significantly outperformed the R^{2}SP for SNRs of 20 and 10 dB (hence the dashed underline of the LSM in Table 6.23). The R^{2}SP did

not give a significant improvement in performance when compared to the TD-ELM, TRM, OSBP-ESN or the LSTM.

Analysis of the performance of the R^2SP's performance when applied to the concrete datasets shows that, as stated above, it significantly outperforms an ESN on both datasets, and also significantly outperforms the TD-ELM when applied to the laboratory-controlled mesh dataset. In the case of the Temple Moor dataset, the TD-ELM provides a significant improvement in performance. The TD-ELM also significantly outperformed the R^2SP when applied to the fourth order polynomial and the spoken digit recognition tasks.

This chapter has presented the results of applying an ESN, R^2SP and TD-ELM to the case studies presented in the previous chapter. For some of these case studies, comparisons against RANNs trained using conventional algorithms and state-of-the-art architectures reported in the literature have also been made. In all case studies, the R^2SP shows an overall significant improvement in performance when compared to an ESN, as well as at least offering comparable performance against the state-of-the-art. In all but one case, the R^2SP was significantly outperformed by the TD-ELM architecture. These results are discussed in more detail in the following chapter, along with conjectures on how the R^2SP provides a significant improvement in performance for those case studies where it gives best performance, and in the case studies where it does not provide the best performance, reasons for this are also discussed.

Chapter 7

Discussion

7.1 Introduction

The previous chapter presented the empirical results obtained when applying various RANN approaches to a variety of artificially created and real-world problems. The reservoir with random static projections (R^2SP) and time delayed extreme learning machine (TD-ELM) approaches introduced in Chapter 4 offered a significant improvement in performance compared to an ESN for the majority of case studies, with the TD-ELM offering the best performance for all of the datasets to which it was applied with the exception of the laboratory controlled mesh dataset. As the trade-off between non-linearity and memory in reservoir computing, particularly ESNs, was the focus of the research, the TD-ELM was not applied to all of the datasets presented previously. The R^2SP offered a significant improvement in performance compared to the ESN for the majority of datasets, with the exception of the extended XOR task where with selected variations of non-linearity and temporal delay the ESN significantly outper-

formed the R^{2}SP. For the extended XOR task the error of both architectures increased drastically with small increases in the non-linearity and temporal delay parameters, indicating the difficulty of the task.

7.2 Training times

Comparison with traditional RANN training algorithms revealed that, as has often been reported in the literature, the reservoir and ELM-based training approaches are much more efficient, and as a result, allow for comparatively easier parameter tuning, which has added to their improved performance. Due to their slow and complicated training procedures, finding the parameters which enable best performance for the traditional RANNs can be difficult often taking many hours to train one network with a set of parameters which, as well as problems such as local minima and vanishing gradients affecting convergence, hindered their performance on the different cases studied. Whilst every effort has been made to optimise the training parameters of these networks, due to the slow training times further optimisation was prohibitive. It is therefore likely that better parameter settings for these ANNs do exist, but given the training times of the different approaches they were difficult to find. The training time of the R^{2}SP was longer than for the ESN (as a result of having more output weights to train). The training times of the TD-ELM approach were also longer than the ESN and also longer than the R^{2}SP in all but one case study. The increased training times of the TD-ELM is these cases is due to the increase in the size of the input layer required to present a window of inputs to the network. Despite their increased training times, the TD-ELM and R^{2}SP offered significant improvement in performance in comparison to the ESN for the problems investigated here and were still much faster than the traditional RANN training algorithms.

7.3 Non-linear separation and memory capacity

The previous chapter also showed (through the analysis of an ESN and R^{2}SP when applied to the extended polynomial dataset) that the improvement in performance offered by the R^{2}SP can be credited to its separation of non-linear mapping and MC through the selective use of two memory-less feedforward layers and an ESN-type reservoir in combination. By changing the parameters that control the amount of non-linear separation and memory required by the task, it was shown that the R^{2}SP-reservoir is usually tuned towards maximising its MC: with minimal non-linear mapping required as this can mainly be performed by $stat_1$. Even in the situation where little memory is required by the task, it was shown that the R^{2}SP-reservoir was still normally tuned towards maximal MC, which confirmed that the non-linear separation continued to be mostly performed by $stat_1$. The analysis of the amount of non-linearity and MC of the R^{2}SP-reservoir compared to the ESN's reservoir when applied to the extended polynomial task is a particular illustration of this. The role of $stat_2$ was to facilitate the readout by transforming the reservoir activations onto a larger state space, something which has been shown to improve performance in a previous study [202].

Further analysis of the activations of each layer of the R^{2}SP revealed the architecture's ability to provide an instantaneous reaction to a change in input, via $stat_1$'s neurons, further aided performance. This is in contrast to the neurons of a reservoir which do not react instantly to a change in input, as was shown when presenting the R^{2}SP network with a constant input for a duration of time and then altering this input to a new value. A reservoir is slow to react to a change in input due to its recurrent connections which allow the neuron activations from previous time steps to influence the activations of its neurons at the current time step. While this is a vital character-

istic of a reservoir (as it facilitates its short-term memory) in some cases where fast changing input data is present this may hinder its performance. This problem can be overcome by artificially slowing the rate of change of the dataset down by repeating datapoints, but repeating datapoints increases the size of the dataset which can in turn increase the training times of the network.

The fact that the TD-ELM, which separates memory and non-linear mapping through the use of a sliding window and a memoryless feedforward layer, also outperforms the ESN in all but one case study shows that the antagonistic trade-off using the ESN can be overcome using a similar approach to the novel R^2SP. Comparing both approaches, the TD-ELM's temporal representation of the input data comes from its window of inputs, with its size directly determining the length of its memory. The R^2SP on the other hand, adopts the recurrent technique of capturing temporal information using a recurrent reservoir which has a fading memory of its inputs. For the datasets studied here, it may be that in the majority of cases the TD-ELM's approach of capturing temporal information of its input via its finite memory over the length of its window may be better suited to these tasks. This is especially the case in the fourth order polynomial task where, as a result of the current output being influenced by only the previous output, a window size of two time steps is sufficient to capture all of the temporal information required. As a result of the memory requirements being over a small temporal period, the fading memory of the reservoir is not required for this task and may in fact hinder performance somewhat as older memories of inputs may interfere with the two most recent time steps for which this task required memory of. For one case study, the laboratory controlled mesh, the reservoir-based approaches were shown to outperform the TD-ELM approach. Here it would appear that the use of a network with a fading memory is best suited to this EMAD dataset.

7.4 Network sizes

The size of the reservoir of the ESN and the R^{2}SP-reservoir that enabled good performance were found using an empirical search which involved increasing the number of neurons by 50 until the increase in performance appeared to plateau. Once this number was found, the reservoirs for both architectures were set at this size. The size two additional layers of the R^{2}SP were set to contain the same number of neurons as the R^{2}SP-reservoir. As a result of adopting this approach, the R^{2}SP's total number of hidden layer neurons was three times that of the ESN's reservoir. The number of neurons in the hidden layer of TD-ELM was found to be best when using a hidden layer size similar to the total number of neurons in the R^{2}SP hidden layer. One may argue that the significant improvement in performance of the TD-ELM and R^{2}SP could be down to the increase in their computational power caused by a larger number of neurons that each architecture possessed. However, the fact that the TD-ELM and R^{2}SP neurons have many more neurons at their disposal compared to the ESN is not the reason for their superior performance. The reason that improved performance was obtained using these two approaches is due to a relaxing of the antagonistic trade-off between an ESN's non-linear separation and memory mapping of its input data which, as this thesis has shown, is particularly prevalent when this input data is fast changing and highly non-linear. Other work, for example Schrauwen and Büsing [249], used reservoir sizes larger (a reservoir containing 1,200 neurons in the work of Schrauwen and Büsing [249]) than those investigated in the study here, where much smaller WERs were obtained using these larger networks. While improved performance for this task can undoubtedly be achieved through the use of larger networks, a comparison against reservoirs with similar computational power was the focus of this study.

Alternative performance comparison strategies are possible: one alternative approach could have been to specify that all architectures had the same number of trainable parameters (i.e. output weights). Using the same number of output weights for the R^{2}SP would have meant a reduction in the number of neurons contained in the R^{2}SP-reservoir, $stat_1$ and $stat_2$ layers. This would have meant that the R^{2}SP-reservoir's computational power would have been reduced due fewer neurons in comparison to the ESN. Initial trials were investigated using this approach, where early indications showed that in the majority of cases containing highly non-linear data, using the same number of trainable parameters does not lead to a significant improvement in performance for the ESN due to the antagonistic trade-off it experiences when processing such datasets.

Pruning of the least significant neurons of the R^{2}SP has suggested that the best topological configuration may not be one in which all three sub-layers of its hidden layer have the same number of neurons as was the strategy used in the research here. Pruning revealed that the best approach may be to set the R^{2}SP-reservoir to contain the highest number of neurons, followed by smaller $stat_1$ and $stat_2$, although their best relative sizes were task dependent. Further investigations into the best topological configuration for the R^{2}SP when applied to related tasks is an area for future work.

7.5 Case studies investigated

7.5.1 Fourth order polynomial

For each of the case studies presented, the MC and non-linearities present in the reservoirs of the ESN and the R^{2}SP were calculated. For the fourth order polynomial task it was shown that as a result of the ESN's tuned input scale and spectral radius,

it had a lower MC, higher amounts of non-linearities and a maximum local Lyapunov exponent larger than zero, indicating an unstable reservoir. The R^2SP-reservoir on the other hand, had a much higher MC and lower amounts of non-linearities as well as a local Lyapunov exponent smaller than zero, which indicated a stable network.

7.5.2 Extended polynomial

For the extended polynomial task (and the XOR task discussed in section 7.5.3), the deviation from linearity and MC of both architectures varied depending on how the parameters which controlled non-linearity and MC required by the task were incremented. When increasing the non-linearity and MC requirements of the task simultaneously the MC of the R^2SP-reservoir generally increased, whereas the MC of the ESN's reservoir alternated between high and low values. When the target output contained high amounts of non-linearities and delay the reservoir of the ESN had a much lower MC compared to the R^2SP. At these high values of non-linearities and delay in the target output data, the amount of non-linearities contained within the ESN were high, hence the lower MC at these values.

When only increasing the delay in the target output, the MC and deviation from linearity of the two architectures were similar. On both cases reservoirs were tuned towards their maximal MC, as was shown by the values of input scale and spectral radius which enabled best performance (in all cases the spectral radius never exceeded unity, while the input scale rarely did). Although the MC of the two architectures was similar when increasing the delay in the input data, the R^2SP-reservoir had the larger MC of the two architectures in 11 of the 15 values of the delay tested. In these cases the increased MC of the R^2SP may have aided performance, although the difference in performance between the architectures is similar for all values of the delay, which

suggests that the instantaneous non-linear projection provided by $stat_1$ may be the principal aid to performance here. The values of the deviation from linearity for both architectures generally increased with an increase in the delay, which was caused by a spectral radius value approaching unity which was required to increase the MC of both architectures. These values were not as high as when increasing both the non-linearity and delay of the dataset simultaneously however.

The R^{2}SP-reservoir was also tuned towards maximal MC when only the non-linearity of the target output increased, which shows that the majority of the non-linear transformation of the R^{2}SP were performed by $stat_1$.

7.5.3 Extended XOR

The smaller amounts of non-linearities present in the R^{2}SP-reservoir (4 out of 15 cases) versus its higher MC (8 out of 15 cases) was also evident in the majority of cases for the extended XOR task, which aided performance. In one case (where the delay was equal to five time steps) the MC of the R^{2}SP was higher than the ESN but despite this the ESN offered significant improvement in performance compared to the R^{2}SP. Although the errors of the two architectures were significantly different in 13 of the 15 trials, the MC and non-linearities contained in both architectures' reservoirs were similar; however, the generally lower amount of non-linearities in the R^{2}SP-reservoir coupled with its somewhat higher MC in the majority of cases may have helped improve performance significantly.

7.5.4 Spoken digit recognition

The R^2SP also outperformed an ESN when trained and tested on noisy versions of the spoken digit recognition task, where a lower tuned value for the spectral radius which enabled best performance was found, although both values were above unity. As a result of this, the R^2SP had a higher MC, lower amounts of non-linearities present in its R^2SP-reservoir and a lower maximum local Lyapunov exponent. As both architectures obtained a maximum local Lyapunov exponent larger than zero, both can be said to be unstable, although the ESN was more so. The same behaviour for both architectures was also observed when training with clean spoken digits and testing on various unseen noisy digits, which was caused by a higher tuned spectral radius values for the ESN. The TD-ELM approach was shown to outperform the R^2SP and ESN approaches for test digits over all SNRs.

This thesis has also compared the performance of an ESN, R^2SP and TD-ELM with a number of other approaches which have been reported in the literature and applied to the same spoken digit recognition task delivering mixed results. When tested on unseen clean samples, the OSBP reservoir training approach offered by far the best result out of the reservoir-based architectures, but due to the added complexity of training the input-to-reservoir weights longer training times are common. The TD-ELM's performance was the closest out of the remaining reservoir and ELM based approaches, while the R^2SP offered comparable performance to the LSM and LSTM approaches. In the case of using spoken digits corrupted with various amounts of noise as the test data, the TD-ELM was found to offer the best performance for two levels of noise (30 dB and 20 dB), while the LSM gave the best performance when applied to the test utterances corrupted with an SNR of 10 dB. In the case of the nosiest utterances

the TD-ELM was second to the TRM which offered by far the best classification error. At this level of noise the ESN, TD-ELM and R^{2}SP were shown to outperform a CRBM and a state-of-the-art HMM.

7.5.5 Reinforced concrete datasets

The performance of an ESN, R^{2}SP and TD-ELM when applied to two EMAD datasets were one of the principal foci of this research. The R^{2}SP was shown to significantly outperform the other two approaches for the laboratory controlled mesh dataset. Whereas the TD-ELM was shown to significantly outperform the other two architectures for the real-world concrete dataset. In the case of the laboratory controlled mesh dataset, where the ground truth of the location of each defect was known, the R^{2}SP was shown to give less false positives while a comparable number of defect detections were obtained in comparison to the ESN. The ESN and R^{2}SP were shown to outperform the TD-ELM architecture for this dataset, suggesting that this mesh dataset is best processed using an architecture whose short-term memory is captured using a fading approach rather than the fixed-window approach adopted by the TD-ELM.

The tuned parameters for the ESN and R^{2}SP were very similar for the laboratory controlled mesh task, with a smaller input scale for the ESN giving it a slightly higher MC and deviation from linearity. As the defects present within the mesh dataset tended not to be present for many time-steps, the lower MC of the R^{2}SP does not hinder performance and the instantaneous non-linear transformation performed by $stat_1$ also aided performance. The best window size for the TD-ELM was found to be 15 datapoints, which resulted in slower training times when compared to both the ESN and R^{2}SP approaches.

The tuned parameters of the ESN and R^2SP when applied to the Temple Moor dataset differed much more than was the case for the mesh dataset. The input scale and spectral radius of the ESN were extremely high which was probably caused by the high amounts of non-linearities contained within the dataset. Recall that the two EMAD datasets were collected from structures with a different configuration (rebar mesh versus single steel tendons), which may explain the large change in parameters which enabled best performance. As a result of these high values, the MC of the ESN was low when compared to the R^2SP-reservoir and, as expected, the ESN's deviation from linearity was very high. The ESN also appeared to be unstable as a result of its parameters, as indicated by a maximal local Lyapunov exponent which was greater than zero.

The reason for the TD-ELM and R^2SP delivering improved performance when analysing the Temple Moor dataset seems to be the dataset containing more non-linearities than the data collected from the laboratory controlled concrete slab: the EMAD signatures for which, in comparison to the Temple Moor data, were quite flat for the majority of the the scan lines. As the Temple Moor dataset is highly non-linear, the ESN is outperformed by the other two approaches as these approaches are better able to overcome the antagonistic trade-off between non-linear transformation of the input data and the ability to recall inputs from the past. The fixed memory of the TD-ELM gives slightly better performance when compared to the fading memory used by the R^2SP, suggesting that data collected from buildings containing single steel tendons may be best suited for processing using the TD-ELM.

While the improvement in performance of the TD-ELM compared to the other two approaches when applied to the Temple Moor dataset was not huge, it was still signifi-

cant and any improvement in results is desirable as more defective zones are detected. Detecting a higher proportion of defective areas could potentially prevent structural collapses and may reduce repair costs as the earlier a defect is detected the less it generally costs to repair. The R^2SP gives the fewest false alarms as indicated by its higher specificity. This is desirable as false alarms are costly since they require needless inspection which can involve the removal of the concrete from a healthy section of reinforced concrete. This not only then requires repair, but also increases the chance of a previously healthy section deteriorating more quickly in the future.

7.6 Reservoir dynamics when applied to all datasets

To summarise the dynamics of both reservoir based techniques, Table 7.1 below shows the dynamics of the ESN and R^2SP when applied to all of the datasets presented in this study, in particular which architecture's reservoir had the lowest $\widetilde{LLE}_{max}$, lowest δ_ϕ and highest MC.

Table 7.1: The dynamics of the ESN and R^{2}SP when applied to all of the datasets investigated in this study showing which architecture had the reservoir with the smallest maximal local Lyapunov exponents ($\widetilde{LLE}_{max}$) i.e. the most stable characteristics, lowest amounts of deviation from linearity (δ_ϕ) and the highest memory capacity (MC).

Training dataset	**Smallest** $\widetilde{LLE}_{max}$	**Smallest** δ_ϕ	**Highest MC**
4th order poly	R^{2}SP	R^{2}SP	R^{2}SP
Extended poly increasing p and d	n/a	R^{2}SP (13/19 cases)	R^{2}SP (10/19 cases)
Extended poly increasing d	n/a	ESN (9/18 cases)	R^{2}SP (11/18 cases)
Extended poly increasing p	n/a	R^{2}SP (18/18 cases)	R^{2}SP (17/18 cases)
Extended XOR	n/a	R^{2}SP (7/15 cases)	R^{2}SP (8/15 cases)
Speech noisy	R^{2}SP	R^{2}SP	R^{2}SP
Speech clean	R^{2}SP	R^{2}SP	R^{2}SP
Mesh	R^{2}SP	R^{2}SP	ESN
Temple Moor	R^{2}SP	R^{2}SP	R^{2}SP

As Table 7.1 shows, for the majority of datasets the R^{2}SP-reservoir had a higher MC as a result of a reduced requirement to process non-linearities, as indicated by lower values of δ_ϕ. In just one case (increasing the value of temporal extent of short-term memory, defined by d, in the extended polynomial task) the ESN had the majority of smaller deviation from linearity values. While a higher number of non-linearities may

be seen as an unwanted characteristic of the R^{2}SP-reservoir (as high values decrease its MC), in fact the opposite is true. As was shown earlier in Figures 3.14 and 3.15 of Chapter 3, for the MC of a reservoir to increase, its deviation from linearity value must also increase slightly. This is the case here for the R^{2}SP when increasing the delay of the extended polynomial task which, as shown in Table 7.1, possessed a larger MC. Only when the non-linearities, as measured by the deviation from linearity score, reach a threshold does the MC decrease. As a result of their smaller values for their best tuned parameters, in particular their input scale and spectral radius, the R^{2}SP-reservoir layers have a smaller maximal $\widetilde{LLE}_{max}$ than the ESNs: an indication of the ESN's greater instability, which would contribute to their inferior performance. The ESN was found to have a larger MC than the R^{2}SP-reservoir in one case: when trained on the controlled mesh dataset. This was caused as a result of a smaller input scale required by the ESN. Despite this increase in memory capacity of the ESN's reservoir, it was still significantly outperformed overall by the R^{2}SP network for this task. The short and fast changing EMAD signatures offer an explanation for this.

7.7 Research hypotheses for research question 3 and contribution 3

This section summarises the reasons why the R^{2}SP architecture outperforms other architectures when applied to the datasets investigated in this research. Some of these reasons are conjectures based on theory on behalf of the author, and require further work to validate. The table below summarises the reasons/conjectures for the R^{2}SP's performance against other architectures to which it was compared.

Table 7.2: The different architectures the R^{2}SP's performance was compared against when applied to various datasets presented in this research. Here the hypotheses are given for why the R^{2}SP outperforms the architectures it does, and where it does not provide the best performance hypotheses are also given. Where the R^{2}SP provides best performance, the task is underlined.

Architecture R^{2}SP compared against	**Dataset(s)**	**Reason(s) for superior performance**
ESN	All datasets[1]	The R^{2}SP overcomes the antagonism between non-linear transformation and short-term memory present within an ESN as shown in Chapter 6.
RANNs trained using conventional approaches (EN, LRN, DTDNN, EN-BPTT and EN-RTRL)	Fourth order polynomial	Using a simple and efficient training algorithm which overcomes problems associated with conventional RANN training algorithms such as vanishing gradients and local minima [40].

Continued on Next Page...

[1] In 11 of the 15 cases for the extended delayed XOR task

Table 7.2 – Continued

Architecture R^2SP compared against	**Dataset(s)**	**Reason(s) for superior performance**
RANNs trained using conventional approaches (EN, LRN, DTDNN)	Spoken digit recognition	Using a simple and efficient training algorithm which overcomes problems associated with conventional RANN training algorithms such as vanishing gradients and local minima [40].
OSBP-ESN	Spoken digit recognition	The OSBP-ESN outperformed the R^2SP due to the additional training of the input-to-reservoir weights using a one step backpropagation through time training algorithm.
RLS-ESN	Spoken digit recognition	The R^2SP overcomes the antagonism between non-linear transformation and short-term memory present within an ESN as shown in Chapter 6.

Continued on Next Page...

Table 7.2 – Continued

Architecture R^2SP compared against	**Dataset(s)**	**Reason(s) for superior performance**
LSM	Spoken digit recognition	The R^2SP gave best performance for an SNR of 30 dB, with the LSM giving better performance for SNRs of 20 and 10 dB. This is perhaps due to the increase in computational power of an LSM compared to an ESN as shown in Maass [199].
TRM	Spoken digit recognition	The TRM outperformed the R^2SP when classifying utterances corrupted using an SNR of 5 dB. This could be a result of the TRM containing more reservoir neurons (1,200) in comparison to the R^2SP (450 neurons), although further work is required to validate this point.

Continued on Next Page...

Table 7.2 – Continued

Architecture R^2SP compared against	**Dataset(s)**	**Reason(s) for superior performance**
CRBM	Spoken digit recognition	The R^2SP overcomes the need to specify a memory depth as is the case with the CRBM as its reservoir has a fading memory. A large memory depth leads to a large number of weights to be learned which slows down training. It may be that for the CRBM's performance, a small memory was used as to make training feasible which may have been insufficient to capture the memory requirements of the task.

Continued on Next Page...

Table 7.2 – Continued

Architecture R^2SP compared against	**Dataset(s)**	**Reason(s) for superior performance**
HMM	Spoken digit recognition	ANN approaches are known to be more noise robust than HMMs as was shown in the case of ESNs [252, 253], LSMs [247] and LSTM architectures [337], which could be attributed to the R^2SP's superior performance.
LSTM	Spoken digit recognition	The R^2SP achieved a comparable error rate to the LSTM when tested on clean spoken utterances and further data is needed to determine the significance of these results. The LSTM has been shown to be able to deal with long time delays in data (hence a large MC) [337] as is the case with reservoirs, although the two architectures achieve this in different ways.

Continued on Next Page...

Table 7.2 – Continued

Architecture R^2SP compared against	**Dataset(s)**	**Reason(s) for superior performance**
TD-ELM	Fourth order polynomial, spoken digit recognition, laboratory controlled mesh, Temple Moor	The R^2SP was significantly outperformed for three of the four datasets where it was compared against the TD-ELM. The reason for this could be due to the memory requirements of these datasets being sufficiently captured using the time-windowing approach of the TD-ELM. This is particularly the case for the fourth order polynomial where a window size of two time steps is sufficient. In the case of the laboratory controlled mesh, the R^2SP's superior performance is attributed to the dataset, which is less non-linear (in comparison to the Temple Moor data) and the defects are present over a longer period of time, which may suit the fading memory of the R^2SP-reservoir.

7.8 Pruning of reservoir neurons

The pruning algorithm introduced in Chapter 4 helped to improve performance of the R^{2}SP when applied to the fourth order polynomial, spoken digit recognition, and laboratory controlled mesh datasets. This improved performance was achieved as a result of the further regularisation that the pruning process offered: removing the R^{2}SP neurons and their output weights which were not required by each task. While this improvement in most cases was only slight, the performance of the network when tested with noisy spoken digits was improved by roughly 5.3%, which is much higher than the 1% improvement achieved on the other test datasets.

In addition to sometimes slight improvements in performance, pruning also provided further insight into the otherwise opaque way in which the networks use their computational resources to address each problem domain. For all case study tasks investigated using pruning, the optimal R^{2}SP-reservoir layer was always the largest of the three layers in a particular pruned R^{2}SP network, while $stat_1$ had the second highest number of neurons in three out of the four test cases. This revealed that the MC capabilities of the R^{2}SP-reservoir were vital in all tasks: with the exception of the fourth order polynomial where little MC was required and most of the $stat_1$ and $stat_2$ neurons were removed. This is unexpected, as one might have expected the majority of R^{2}SP-reservoir neurons to be pruned as a result of the little short-term memory input-recall required from the fourth order polynomial task. It nonetheless appears that the R^{2}SP-reservoir performed more non-linear mapping for this task as indicated by its comparatively high number of remaining neurons. This is likely to be a result

of its higher than unity tuned input scale of 2.5 which would have shifted its neurons' working points towards their non-linear regions. Had an input scale of unity or below been used for the R^{2}SP-reservoir and a higher input scale for $stat_1$ and $stat_2$, the R^{2}SP-reservoir neurons may have been more linear and as a result it might be expected that the majority of neurons would have been removed from the R^{2}SP-reservoir layer.

While pruning provided an increase in performance as well as added insights into the most useful parts of the R^{2}SP for a given task, it was very computationally demanding due to the recalculation of output weights after each neuron removal. The pruning algorithm is also task dependent, meaning that a pruned R^{2}SP network whose size was optimised for one task, would not offer the same improvement in performance for another task. Therefore the pruning procedure would need to be applied to each dataset. As a result of its task dependency and computational demands, pruning is best suited for investigative studies rather than operationally oriented performance dependent studies where fast training times are required.

With the R^{2}SP the size of its layers increased as the difficulty of the task increased. The simple fourth order polynomial task allowed the majority of the R^{2}SP neurons to be removed, followed by the tests with unseen clean spoken digits. When tested with noisy digits, the number of neurons remaining after pruning increased further still which is probably due to the fact that the task was more difficult and, hence, more neurons needed to be retained. If this is the case, then the average number of neurons removed from the R^{2}SP when tested on the unseen mesh scan lines would suggest that this was the most difficult case study task out of the case studies investigated, as this had the highest number of neurons remaining with only 23 out of 600 removed.

7.9 Summary

This thesis has shown empirically that the novel R^{2}SP architecture significantly outperforms an ESN in nearly all of the case studies investigated. When applied to a case study, the TD-ELM was further shown to significantly outperform both architectures with the exception of the laboratory controlled mesh EMAD dataset, where the R^{2}SP gave the best performance. The improvement in performance of both architectures is a result of separating the non-linear mapping of the input from the ability to recall inputs from the previous time step, a task which ESNs can find difficult to achieve. This is due to the antagonism that arises between the settings of its parameters which allow best performance: simultaneously high amounts of both are unachievable. Analysis of the dynamics of the reservoirs of the ESN and R^{2}SP architectures revealed that the R^{2}SP-reservoir generally contained less non-linear neurons, as the non-linear processing operations were performed by the static feedforward layers, thereby allowing the R^{2}SP-reservoir to possess a higher MC when compared to an ESN architecture. In most of the case studies investigated, the tuned parameters of the R^{2}SP were within their recommended operational range (i.e. not greater than unity), with only a few examples of the input scale and spectral radius values higher than unity. In comparison, the parameters of the ESN that provided best performance were often higher than those of the R^{2}SP and for case studies such as the Temple Moor dataset they were very high, having a detrimental effect on its ability to recall previous inputs. This problem was also seen to arise when analysing the stability of the ESN where large values for its input scale and spectral radius contributed to network instabilities.

The improvement in performance offered by the R^{2}SP also derives from its ability to provide an instantaneous non-linear transformation of the input data via its $stat_1$ layer.

As was shown in the previous chapter, the recurrent reservoir neurons are much slower in reacting to a change of input, although they are necessary to enable the reservoir to possess short-term memory.

For most of the datasets presented here, combining an ELM with a moving window of the data, therefore providing it with memory which stretches over the length of the input window, was shown to offer improved performance when compared to both an ESN and R^{2}SP. This again shows that by separating these two task requirements, improved performance can be obtained. The significant improvement in performance of the TD-ELM could be due the datasets studied here, where most appear to be best suited for analysis using an approach with a finite fixed length temporal representation of the mapping between the input and output data, rather than using an approach with a fading temporal representation of the data, as is the case in reservoir computing and R^{2}SP.

For one particular dataset (the laboratory controlled mesh case study), the R^{2}SP offered a significant improvement in performance compared to the other two architectures. This appears to be a result of the fading memory provided by the R^{2}SP-reservoir being better suited to capturing the temporal characteristics of the dataset, where the fixed memory of the TD-ELM appears to be ill-suited.

The superior performance of the R^{2}SP architecture when compared to the ESN can be further enhanced by pruning the least significant output weights of the R^{2}SP: this was also shown to provide added insight into the requirements of the task as well as the dynamics of the network.

Chapter 8

Conclusions

8.1 Overall summary

This research has empirically investigated the use of established, recent and one completely novel recurrent neural network architectures for time-series data analysis which, as a result of their ability to learn the mapping between inputs and outputs over a period of time, make them well-suited to data of this kind. Traditionally the use of recurrent neural networks for this analysis has been somewhat limited as a result of their slow and complex training algorithms, which makes very good performance difficult to achieve. A relatively recent innovation to training recurrent neural networks, Reservoir Computing (RC), overcomes these problems by training only the output weights, often using a simple linear regression technique which finds the optimal weights given certain network parameters. Due to the appeal of their simple training procedure, the application of RC to time-series domains has become widespread, despite their relative infancy, where performance matching at least the state-of-the-art has often been reported.

The recurrent neural networks investigated in this thesis were applied to a wide range of datasets, both artificially created and from real-world problems, with an emphasis on the automated detection of potential problem areas in data collected from reinforced concrete structures. The corrosion of the steel reinforcement (rebars) inside reinforced concrete is a large and costly problem for the construction and maintenance industry, as problems are often difficult to detect by expert eye. Left unchecked, the corrosion process can weaken the structure significantly, leading to costly repair bills, and in the worst case scenario, the partial or catastrophic total collapse of the structure. This problem is likely to worsen in the next 10-20 years as more structures approach the end of their serviceable lives.

In this study reinforced concrete structures were assessed using a technique known as ElectroMagnetic Anomaly Detection (EMAD) as the means of data capture. As the name suggests, this approach collects some of the electromagnetic properties of the rebars for analysis, which data then need to be analysed and labelled by expert inspection. The EMAD approach has several advantages over alternative testing techniques as it requires no contact with the rebars, is lightweight and, hence, easy to operate and is not effected by environmental factors such as the water saturation levels of the structure under assessment. The main disadvantage of this approach until now has been the need for expert analysis of the data collected to identify potential areas that require further attention. Until now this could only be performed by a team of human experts which was therefore costly and time-consuming. As each reinforced concrete structure has a slightly different construction configuration, very simple threshold based approaches for this analysis are ill-suited as an automatic solution for this problem (as was shown in a study by Obst et al [268] where threshold-based approaches were unsuitable for anomaly detection in gas concentrations of coal mines). Since small

amounts of corrosion near the surface will create the same signal as large amounts of corrosion deep within the structure, both signals could be undetected by a simple threshold technique. Alternatively using a low threshold would give rise to a high level of false positives, which again is undesirable. As the EMAD data is time-series based, recurrent neural networks, particularly RC approaches, are ideally suited to this task since they offer the ability to classify data based on their learned knowledge of the relationships encoded by the time varying EMAD signatures.

In order to provide training and test EMAD data for the neural networks to analyse, a controlled laboratory experiment was created which comprised a standard reinforcing mesh that was set in concrete. The advantage of using this approach was that the ground truth regarding location and type of each defect was known prior to concrete encasement. This allowed the collection and profiling of each type of defect over the time course of the experiment. Data such as this had been absent from the literature until the work reported here. Ground truth defects containing either corrosion, breaks or a combination of the two were systematically introduced onto the reinforcing mesh prior to concrete encasement. Once trained using this dataset, the neural networks could be applied to structures consisting of a mesh construction with little or no further training required.

The corrosion of the rebars was accelerated over the duration of the experiment by repeatedly soaking the concrete in an aggressive environment for a period of two weeks. After the two week soaking period, the concrete was removed from its corrosive environment to enable it to dry out and allow air and, more importantly, carbon dioxide to penetrate the concrete. The process was then repeated for the duration of the work reported here and is still on-going (see Future Work below). To allow the corrosive

environment and air to penetrate the concrete more easily, the concrete was mixed using a higher than normal water-to-cement ratio that made it more porous. While this ratio is larger than that which is defined by construction industry standards, in principle it facilitated the corrosion process so as to exacerbate the introduced defects so that the time course of the corrosion process could take place in the short space of time available for the experiment. (In some cases water-to-cement ratios similar to the one used here are suspected to have been used when building real-world structures through poor construction practices.)

Corrosion products were also systematically introduced onto the mesh following an established approach. The presence of these corrosion products after heating was confirmed from a pilot study using x-ray diffraction. Due to the accelerated corrosion of the concrete slab, it was scanned using the EMAD on a weekly basis to avoid missing any corrosion processes that would of course occur in real-world structures over a longer time period.

Several preliminary experiments were conducted to establish the scanning procedure that should be adopted in order to observe each defect most clearly. Here it was found that scanning and energising in multiple directions, time permitting, was the best approach to adopt. In most real-world cases on-site where time is often limited, energising and scanning longitudinally is recommended as this gives the clearest signatures for the majority of defects. Using the defects whose signatures were present in the EMAD data, each was labelled accordingly in preparation for its analysis using several supervised recurrent neural networks.

8.2 Contributions of this thesis

Section 1.5 of Chapter 1 outlined three contributions of this research. The first contribution was as follows.

1. "A novel R^2SP architecture which combines a standard RC approach (ESN) with two layers of memoryless feedforward neurons, borrowed from the field of ELMs. These two layers perform a non-linear transformation of the input data and reservoir activations which allow the reservoir to be automatically tuned towards a higher memory capacity, improving performance for highly non-linear time-series datasets."

As the point above states, the main contribution of this work has been the introduction and empirical investigation of a novel RC architecture which was created with the aim of overcoming an antagonistic trade-off that is usually present with the ESN approach when processing highly non-linear time-series data: the trade-off between its non-linear mapping capabilities and ability to simultaneously possess short-term memory. This trade-off often becomes apparent when setting important parameters of each ESN that are commonly tuned using an intensive empirical search. In particular an ESN's input scale and spectral radius can allow it to be tuned towards one capability or the other. Other parameters such as the leak rate and bias can also influence these two capabilities of an ESN's reservoir, although the input scale and spectral radius have the most prominent effects. This has been shown by applying a reservoir to a dataset collected from a real-world concrete structure where, as a result of the high non-linearities present in the data, the well tuned values found using a grid search for the input scale and spectral radius were much higher than the usual recommended values. As a result, a very non-linear ESN was usually created where the working points of its neurons were in the non-linear region of their activation functions. In extreme

cases with very high input scale and spectral radius values, an ESN has reservoir neurons whose activations are saturated at their extremes. This is undesirable since the short-term memory of the reservoir in this state is severely compromised. The two pre-eminent RC capabilities have been shown to be antagonistic: either an ESN can possess longer short-term memory or it can perform high amounts of non-linear transformation. For the EMAD concrete tasks some short-term memory was required due to the time-series nature of the dataset, but the defect signatures, especially for the Temple Moor dataset, were highly non-linear.

8.2.1 Reservoir with Random Static Projections

To address this undesirable antagonism, a new RC architecture, Reservoir with Random Static Projections (R^2SP), was introduced. R^2SP combines an R^2SP-reservoir with two memoryless feedforward layers of neurons. The first memoryless layer, $stat_1$, receives the same input as the R^2SP-reservoir and performs an instantaneous non-linear transformation of the input, while the second memoryless layer, $stat_2$, performs an instantaneous non-linear transformation of the current R^2SP-reservoir neurons' activations. R^2SP was created in order to show that by removing the need for the R^2SP-reservoir to perform all of the non-linear transformation of the input data (as is the case when using an ESN) it can indeed be autonomously tuned towards maximising its memory capacity. The majority of the non-linear transformations of the input data are then be performed by $stat_1$, while $stat_2$ provides an instantaneous non-linear transformation of the R^2SP-reservoir neurons.

The second major contribution of this work, as stated in Chapter 1 is.

2. "When applied to several time-series datasets, overall, the R^{2}SP is shown to outperform an ESN for the all of the datasets. Through the use of reservoir analysis measures, it is shown that the reservoir of the R^{2}SP does indeed have a higher memory capacity as a result of the additional memoryless layers transforming the input data and reservoir activations onto a high dimensional state space."

Comparing the R^{2}SP's performance with an ESN's on several tasks requiring various amounts of non-linearity and short-term memory revealed that the R^{2}SP significantly outperformed the ESN in almost all cases. The only exception was in the case of four different values of the power and delay parameters for an extended XOR task, where the ESN offered a significant improvement in performance for only two of the 15 cases tested. This is likely to be a result of the difficulty of this particular task since the errors of both architectures increased dramatically after a small increase in the non-linearities and delay present in the dataset.

When applied to the EMAD data from the laboratory controlled mesh experiment the R^{2}SP gave less false alarms in comparison to the ESN while giving a similar amount of defect detections, overall giving a significant improvement in performance. For the EMAD data collected from a real-world structure, improved performance was also observed using the R^{2}SP which gave fewer false alarms and detected more defects than the ESN. The R^{2}SP was also shown to provide performance comparable to the state-of-the-art approaches for the speech recognition task, with improved performance shown when applied to noisy unseen test utterances when compared to the ESN.

The dynamics of the ESN's reservoir and R^{2}SP-reservoir used for each of the problem domains were also investigated. Here it was confirmed that for the majority of datasets the R^{2}SP-reservoir was more stable. This was confirmed by analysis of both architectures' maximum local Lyapunov exponents. As a result of containing less non-linearities, the R^{2}SP-reservoir had a higher memory capacity than the ESN.

Separating the non-linear mapping and short-term memory requirements of a task was also investigated by combining an ELM (a memoryless feedforward architecture in which, once again, only the output weights are trained) with a moving window of the input data, thereby providing it with a short-term memory. As the length of this memory was determined by the length of the window, the length of the window became an additional parameter to optimise. This approach has been used in several other studies where it was generally shown to improve performance for time-series datasets [307, 309, 310]. The performance of the TD-ELM compared to the R^{2}SP and ESN was shown to be significantly better for the fourth order polynomial, spoken digit recognition and the real-world EMAD dataset. In the study by Li et al [311], the TD-ELM was also shown to outperform an ESN for some datasets, also confirmed by the work reported here.

The third contribution as outlined in Chapter 1 is outlined below.

3. "Where the R^{2}SP (and the ESN) are outperformed by the TD-ELM approach, it is conjectured that this is due to the dataset under analysis. Where the TD-ELM is best suited appears to be for datasets whose memory requirements of previous inputs can be captured sufficiently using a hard limited memory approach. In this case, the fading memory of the reservoir may inhibit performance. For datasets where a fading memory is required, a reservoir-based approach (such as R^{2}SP) is

better suited as shown by its superior performance."

As the above contribution states, the superior performance of the TD-ELM is intimately related to the datasets it is used to process. As the TD-ELM possesses a finite fixed temporal representation through the use of its fixed-length moving window, it is more suited to datasets whose temporal patterns have a commensurate temporal extent. Datasets such as the fourth order polynomial and the Temple Moor EMAD datasets where the temporal characteristics can be captured using a relatively short time window, may be best suited for analysis using a finite temporal representation. In the case of the fourth order polynomial task for example, only two time steps are required to fully capture the temporal representation required by the target output. The same is applicable to the Temple Moor EMAD data. Here the data appears to be highly non-linear and defects are short lived, typically lasting between 10 to 20 time steps. For these datasets, the fading memory offered by a reservoir may not be required, and may be a hindrance. The ESN and R^2SP's superior performance when applied to the laboratory controlled EMAD data compared to the TD-ELM's performance suggests that the use of a fading memory is best suited for this task. As the two EMAD datasets were collected from two characteristically different rebar configurations, it may be that a reservoir-based technique is best suited to processing data collected from structures that are reinforced using a mesh configuration, while the TD-ELM is best suited to structures which have been reinforced using single lengths of tendon. On the other hand, it could be argued that when processing highly non-linear short-lived defects the TD-ELM is best suited. Further investigation is required to validate which of these hypotheses is true.

The performance of the ESN, R^{2}SP and a TD-ELM approach were also compared to traditional recurrent neural network architectures and training algorithms for the fourth order polynomial and spoken digit recognition tasks. Here the advantage of training only the output weights was very clear. Not only were training times much faster for the reservoir and ELM approaches, but due to their simple training procedure their parameters were easier to tune, resulting in significant improvements in performance. As the training times using the traditional recurrent neural network algorithms were much longer than when training only the reservoir and ELM output weights, parameter optimisation was difficult to achieve and as a result, along with gradient descent convergence problems such as local minima, the performance of the networks trained with these traditional algorithms was much worse than that of the reservoir and ELM based approaches.

In order to investigate the properties of the R^{2}SP further, a novel extended polynomial dataset was introduced which allowed its non-linear separation and short-term memory requirements to be varied, taking inspiration from the extended XOR and the fourth order polynomial tasks that were investigated during this project. Through the analysis of the R^{2}SP when applied to this dataset the instantaneous reaction of $stat_1$'s neurons to changes in the input data was found to contribute to the R^{2}SP's superior performance. As a reservoir contains recurrent connections, it is typically much slower to react to a change in its inputs which for tasks which contain fast changing inputs may be problematic. By altering the non-linear mapping and short-term memory requirements of this task, it was also confirmed that the non-linear transformation of the input data provided by $stat_1$ allowed the R^{2}SP-reservoir to be tuned towards maximal MC. As a result of this the R^{2}SP outperformed an ESN when applied to real-world datasets which were highly non-linear and also required short-term memory.

To summarise the results presented in this thesis, Table 6.23 from Chapter 6 is summarised below.

Table 8.1: The different architectures R^{2}SP's performance was compared against when applied to various datasets. Here the hypothesis that the R^{2}SP offers a significant improvement in performance when compared to a different architecture is true when the architecture it was compared against is underlined.

Architecture R^{2}SP outperforms	**Dataset**	**Error measure**
ESN, TD-ELM, EN, LRN, DTDNN, EN-BPTT and EN-RTRL	Fourth order polynomial (see Table 6.1)	NRMSE
ESN	Extended polynomial (see Tables 6.3, 6.4 and 6.5)	NRMSE
ESN [1]	Extended delayed XOR (see Table 6.6)	NRMSE
ESN	Spoken digit task: trained using noisy utterances (see Table 6.7)	WER
ESN, TD-ELM, EN, LRN, DTDNN, LSM, TRM, CRBM, HMM, OSBP-ESN, RLS-ESN, LSTM	Spoken digit task: trained using clean utterances (see Tables 6.9 and 6.13)	WER

Continued on Next Page...

[1] In 11 of the 15 values of p and d

Table 8.1 – Continued

Architecture R²SP outperforms	**Dataset**	**Error measure**
ESN, TD-ELM	Laboratory controlled mesh (see Tables 6.14 and 6.15)	% defects incorrect (E_{ex_loss}), Sensitivity, Specificity, PPV, NPV, MCC, % specific defects detected
ESN, TD-ELM	Temple Moor (see Tables 6.19 and 6.20)	% defects incorrect (E_{ex_loss}), Sensitivity, Specificity, PPV, NPV, MCC

Pruning of the least significant neurons of the R²SP was also investigated where it was shown to offer a slight improvement in performance, with the highest improvement of 5% achieved when testing the R²SP on noisy spoken test digits. The pruning algorithm also provided some insights into the characteristics of the task. Without exception, the R²SP-reservoir neurons were the most important part of the R²SP when applied to the tasks presented, as shown by the number of neurons remaining in the network that provided optimal performance. For the relatively simple fourth order polynomial task, the majority of the R²SP architecture's neurons were removed whereas for the spoken digit task and the laboratory controlled mesh task comparatively fewer neurons were removed. As the non-linearities and short-term memory requirements present in these

datasets increased, their increased difficulty offers an explanation as to why larger networks were required. Pruning also allowed the best tuned R^{2}SP network to be optimised for each dataset. In the case where the size of the network was reduced dramatically, later networks when applied to the same or similar datasets could be configured with this optimal size, therefore reducing training times even further. Faster training times of the R^{2}SP would be very beneficial when applying them to real-world EMAD data where fast indications of problem areas of a structure could be achieved.

8.2.2 Application to real-world scenarios

The main problem domain of this research has been the automated detection of defects within reinforced concrete structures. With the empirical results obtained during this research, the potential use of the R^{2}SP and TD-ELM approaches to automatically detect defects whilst on-site is huge. For some types of concrete reinforcement where the data may not be too highly non-linear, it appears that the R^{2}SP followed by the ESN are perhaps the best approaches to use for automated defect detection. For more highly non-linear datasets (i.e. collected from steel tendon reinforced structures), the R^{2}SP and TD-ELM approaches appear to be best suited. Not only do these architectures inherit the fast and relatively simple training procedures of RC and ELMs, but they also overcome the antagonism present between an ESN reservoir's non-linear mapping capabilities and its capacity to recall previous inputs, giving improved performance for datasets which are highly non-linear. While further analysis of datasets collected from structures with different reinforcement configurations will certainly be of interest, these approaches could be used to complement one another. Depending on the configuration of the structure and the characteristics of the data collected, the R^{2}SP could be used to give a fast and relatively accurate assessment of a structure's

condition on-site. In the presence of highly non-linear data, the TD-ELM could be used for further analysis once the on-site visit is complete, in order to provide further insight into a structure's condition (where a small window enables best performance, the TD-ELM could be given preference over the R^2SP, as faster training times and increased accuracy can obtained, as shown for the Temple Moor dataset). Using these approaches could therefore improve the current lead-time for the analysis of the data which can currently take several days. This will reduce costs and improve the efficiency of any industrial scale EMAD process, whilst maintaining an acceptable and systematic level of performance. This has the potential to improve the service life of the reinforced concrete infrastructure on which we heavily depend.

8.3 Future work

Some of the defects introduced onto the laboratory controlled mesh were no longer detectable from their EMAD signatures after they had been encased in concrete, probably a result of the formation of a chemical passive layer between the concrete and the rebars, as well as the increased distance between the rebars and the EMAD probe. At the time of the design of the experiment this was unforeseen. The majority of defects that were visible experienced little change in their defect signatures, apart from a few cases where their signature became stronger and, as a result, could be detected from further away. Given the time course of the experiment, the corrosion acceleration approach was perhaps not aggressive enough to create and worsen corrosion processes that typically take between 15 and 50 years to occur in the real-world. The exacerbation of the corrosive areas could have been re-established after concrete encasement and speeded up further by inducing an electric current onto the mesh to create a cathode and anode, but at the time of experimental design this particular approach was not

included. The induction of an electric current has since been implemented onto the concrete slab and further EMAD data collection is currently ongoing.

Due to the evidence summarised by this thesis, SciSite Ltd intend to incorporate an R^2SP network into a new version of the EMAD to collect and analyse data. This will indicate to the EMAD operator, whilst on-site, potential areas of concrete containing defects that require further investigation: either in real time or shortly after the area has been scanned. The TD-ELM could then be used off-site to provide any further information not provided by the R^2SP for structural configurations where a fading memory is not well-suited. If implemented successfully, this approach will improve the efficiency of the EMAD whilst maintaining an acceptable detection rate which could ultimately reduce the time and cost associated with on-site surveys and improve the health of structures in the built environment.

As the behaviour of reservoirs when processing highly non-linear time-series datasets was the main focus of this research, the TD-ELM approach was not applied to all of the datasets presented here. To enable further comparison between the TD-ELM, R^2SP and ESN, it could also be applied to the extended polynomial and XOR tasks. It would be interesting to analyse the performance of the TD-ELM here, where the length of the input window could equal the delay in the target output of the extended polynomial and extended XOR tasks. The TD-ELM may also offer a significant improvement in performance when applied to these two tasks since using a finite temporal representation of the input data gave superior performance for tasks whose temporal characteristics could be captured using a fixed memory. Further to this point, other datasets which require a fading temporal representation should also be investigated. While it is difficult to determine these characteristics of a dataset prior to processing,

benchmark tasks such as the NARMA dataset, with the recursive nature of its output, would probably be suited to a system which has a fading memory, such as an ESN. A recent attempt to bridge the gap between these two temporal representation approaches was proposed by Hermans and Schrauwen [343]. In their work, a Least Squares Support Vector Machine was trained using a window of the inputs which was weighted, essentially increasing the influence of the inputs that were most important for best performance. Their approach investigated several datasets whereby reducing the influence of the less important inputs; improved performance was obtained for certain tasks. A comparison of the TD-ELM and R^{2}SP's performance when applied to these datasets would be an interesting area for future work.

Another approach that could offer a solution to the trade-off between non-linearity and short-term memory in ESNs might be to adopt an ensemble architecture containing many pools of reservoirs each with their own parameters. Using this approach, some reservoirs could be created to maximise their short-term memory, while others could be tuned towards transforming highly non-linear data. A comparison between this approach and the R^{2}SP and TD-ELM would be interesting, although these two architectures could still offer improved performance as a result of the instantaneous reaction of the neurons in $stat_1$ and the hidden layer to a change of input respectively, which for tasks with fast changing data has been shown to be advantageous. Using the ensemble approach would require more parameters to be optimised however, which may increase training times.

Improvements in performance, as well as the added insights into both the task and the most useful R^{2}SP neurons, provided by MRSR-based pruning made this a worthwhile exercise for the research reported in this thesis. Other pruning approaches rather

than the brute force algorithm presented here, such as the approaches adopted in the ELM community to optimise the size of a network, could be investigated as future work in order to speed up the pruning process which currently is very computationally demanding. Pruning or the addition of neurons to the TD-ELM should also be investigated, to investigate if improved performance can be achieved, as was the case with the R^2SP. Further investigation into pruning of the R^2SP when applied to the fourth order polynomial could also be a focus for future work, since more R^2SP-reservoir neurons might be expected to be removed from the network if an R^2SP-reservoir whose neurons' working points operate around their linear regions arose rather than their non-linear regions which was observed in this work.

With the exception of the spoken digit recognition task, the benchmark datasets used in this research have rarely been investigated in other studies. Out of these less used benchmarks, only the extended XOR problem had been investigated previously using an ESN. Here the R^2SP offers a significant improvement in performance for 11 of the 15 different values of the non-linearities and delay present in this dataset. To further validate the performance of the new R^2SP architecture, studies of the performance of other machine learning data processing techniques when applied to the datasets studied above is to be encouraged. In addition, unsupervised approaches to solving these problems may be of interest as their capability of clustering data based on its underlying characteristics may be well suited for the detection and classification of defects within EMAD data. As no unsupervised techniques were studied in this work, a comparison against the performance of the networks reported here would also be of interest.

Further analysis into the effects of the size of the reservoir in the R^2SP is ongoing work. In the research reported here, the size of the ESN's reservoir and the R^2SP-reservoir were the same for each case study. While it could be argued that the R^2SP outperforms the ESN because of its additional number of neurons (contained in the two feedforward layers), early investigations show that this is not the case for the majority of highly non-linear datasets. Rather as shown here, the R^2SP's improvement in performance is instead a result of overcoming the antagonistic trade-off found in an ESN when analysing highly non-linear time-series data.

Finally, the analysis of the performance of the R^2SP and TD-ELM approaches when applied to other challenging datasets with high amounts of non-linearities and long delays would be an interesting focus for future work. One such dataset has recently been collected from a reinforced concrete bridge through collaborative work with the National Physical Laboratory and this is planned for future analysis. The work presented here has shown that the novel R^2SP approach can significantly outperform an ESN when applied to the EMAD datasets, although the TD-ELM can offer improved performance for particularly highly non-linear datasets where a finite, rather than fading, memory of previous inputs seems to work best. When applied to another EMAD dataset, the fading memory of a reservoir meant that the R^2SP approach offered the best performance, followed by the ESN. Further comparison between the architectures when applied to other datasets, would be useful to further build upon these findings.

References

[1] D.H. Campbell and R.L. Folk. Ancient egyptian pyramids - concrete or rock. *Concrete International*, 13(8):28 & 30–39, 1991.

[2] J.M. Vanderley. On the sustainability of the concrete. *Industry and Environment*, 26(2–3):62–63, 2003.

[3] The Concrete Centre. The concrete industry sustainability performance report. Technical report, Riverside House, 4 Meadows Business Park, Station Approach, Blackwater, Camberley, Surrey GU17 9AB, 2009.

[4] World Business Council for Sustainable Development. Cement sustainability initiative. Executive brief, 4, chemin de Conches, 1231 Conches-Geneva, Switzerland, March 2010.

[5] D. Moore. *The Roman Pantheon: The Triumph of Concrete.* `http://www.romanconcrete.com/index.htm`, accessed 01/05/10.

[6] IISI Committee on Economic Studies. Steel statistical yearbook 2007. Technical report, International Iron and Steel Institute, Brussels, 2007.

[7] G.H. Koch, M.P.H. Brongers, N.G. Thompson, Y.P. Virmani, and J.H. Payer. Corrosion costs and preventive strategies in the united states. Technical Report PUBLICATION NO. FHWA-RD-01-156, NACE International, July 2002.

[8] Residual life models for concrete repair - assessment of the concrete repair process. Technical report, BRE Client Report, 2002.

[9] S. Matthews and J. Morlidge. What's wrong: Concrete repair - solution or problem? In *Concrete Solutions*, pages 3–10, 2006.

[10] J.P. Broomfield. *Corrosion of Steel in Concrete*. Taylor & Francis, Oxon, 1st edition, 1997.

[11] E. Boulanger. Study of the interface between pure steel and cement in reinforced concrete. Ecole Nationale Superieure de Ceramique Industrielle, 2009.

[12] N.G. Shrive. Intelligent structural health monitoring: a civil engineering perspective. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1973–1977, 2005.

[13] R.J. Woodward and F.W. Williams. Collapse of Ynes-y-Gwas bridge, West Glamorgan. In *Proceedings of the Institution of Civil Engineers Part I*, volume 84, pages 635–669, 1988.

[14] B. Mathy, P. Demars, F. Roisin, and M. Wouters. Investigation and strengthening study of twenty damaged bridges: A belgium case history. In *Proceedings of the 3rd International Conference on Bridge Management*, pages 658–666, London, 1996.

[15] J.G.M. Wood. Pipers Row car park, Wolverhamption. Quantitative Study of the Causes of the Partial Collapse on 20th March 1997. Technical report, Health and Safety Executive (HSE), 1997.

[16] Bossard Group. Swimming pool roof collapse. Technical report, Steinhauserstrasse 70, 6301 Zug, Switzerland. `http://doyouknow.dk/Default.aspx?ID=92`, accessed 27/10/10.

[17] ASTM C-876-91. Standard test method for half-cell potentials of uncoated reinforcing steel in concrete. Technical report, ASTM International, 1999.

[18] Wagramer Strasse. Guidebook on non-destructive testing of concrete structures, 2002.

[19] P.W. Haycock and R. Lambert. Rust detection - The Scisite Method. *Concrete*, 42:42–43, 2008.

[20] J.B. Butcher, M. Lion, C.R. Day, P.W. Haycock, M.J. Hocking, and S. Bladon. A low frequency electromagnetic probe for detection of corrosion in steel-reinforced concrete. In M. Grantham, C. Majorana, and S. Valentina, editors, *Concrete Solutions*, pages 417–424. CRC Press, 2009.

[21] S.L. Sherratt. *Characterisation of iron oxide corrosion products using impulse ferromagnetic resonance*. PhD thesis, Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM), Keele University, Staffordshire, ST5 5BG, UK, 2010.

[22] H. Scheel and B. Hillemeier. Location of prestressing steel fractures in concrete. *Journal of Materials in Civil Engineering*, 15(3):228–234, 2003.

[23] H. Scheel and B. Hillemeier. Fast location of prestressing steel fractures in bridge decks and parking lots. In *International Symposium on Non-Destructive Testing in Civil Engineering (NDE-CE 2003)*, 2003.

[24] B. Hillemeier and A. Walther. *Advances in Construction Materials*, chapter Fast Non-Destructive Localisation of Prestressing Steel Fractures in Post-Tensioned Concrete Bridges, pages 563–574. Number Part VII. Springer Berlin Heidelberg, 2007.

[25] Y. Li-Jian, M. Feng-Ming, and G. Song-Wie. Study on intelligent quantitative recognition of defect in pipeline magnetic flux leakage inspection. In *12th Asia–Pacific Conference on NDT (A–PCNDT)*, 2006.

[26] A. Sadr and S. Ehteram. A new pattern recognition technique in non-destructive testing by use of linear discriminant analysis. *Modern Applied Science*, 3(5):118–126, 2009.

[27] A. Sadr and S. Ehteram. Intelligent MFL defect detection algorithm equipped by linear discriminate analysis. In *Defektoskopie*, pages 305–312, 2008. Available from www.ndt.net/article/v13n06/ehteram1.pdf, accessed 14/07/08.

[28] A. Sadr and S. Ehteram. Robust defect detection algorithm based classification on features extracted from MFL signals. *NDT.net - The e-Journal of Nondestructive Testing*, 2008. Accessed 14/07/08.

[29] A. Sadr and S. Ehteram. Intelligent defect recognition from magnetic flux leakage inspection. *NDT.net - The e-Journal of Nondestructive Testing*, 2008. Accessed 14/07/08.

[30] R. Christen and A. Bergamini. Automatic flaw detection in NDE signals using a panel of neural networks. *NDT&E International*, 39(7):547–553, 2006.

[31] B. Hillemeier and H. Scheel. Non–Destructive location of prestressing steel fractures in post–tensioned and prestressed concrete. In *Transportation research board (TRB) Committee A2C03 – Concrete Bridges*, pages 1–11, Washington D.C., USA, 2002.

[32] K. Kim and W.B. Lee. Stock market prediction using artificial neural networks with optimal feature transformation. *Neural Computing & Applications*, 13:255–260, 2004.

[33] K. Müller, M. Krauledat, G. Dornhege, G. Curio, and B. Blankertz. Machine learning techniques for brain-computer interfaces. *Biomedical Engineering*, 49(1):11–22, 2004.

[34] A. Ganapathiraju, J.E. Hamaker, and J. Picone. Applications of support vector machines to speech recognition. *Signal Processing, IEEE Transactions on*, 52(8):2348 – 2355, 2004.

[35] Y.H. Kim and F.L. Lewis. Neural network output feedback control of robot manipulators. *Robotics and Automation, IEEE Transactions on*, 15(2):301 –309, April 1999.

[36] T. Yairi, Y. Kawahara, R. Fujimaki, Y. Sato, and K. Machida. Telemetry-mining: A machine learning approach to anomaly detection and fault diagnosis for space systems. In *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, pages 466–476, 2006.

[37] J.R. Bellegarda. Unsupervised document clustering using multi-resolution latent semantic density analysis. In *2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 361 –366, 2010.

[38] G.M. Shephard and C. Koch. *The synaptic organisation of the brain*, chapter Introduction to synaptic circuits, pages 3–31. Oxford University Press, New York, 1990.

[39] H. Skolimowski. *The theatre of the mind: evolution in the sensitive cosmos*. Quest books, 1984.

[40] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, USA, 1999.

[41] M.F. Bear, B.W. Connors, and M.A. Paradiso. *Neuroscience Exploring the Brain*. Lippincott Williams & Wilkins, Baltimore, USA, third edition, 2007.

[42] W. Gerstner and W. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

[43] V.F. Turchin. *The Phenomenon of Science: a cybernetic approach to human evolution*. Columbia University Press, 1977. ISBN 0-231-03983-2.

[44] D.H. Hubel. *Eye, Brain and Vision*. Scientific American Library, 1988.

[45] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[46] F. Rosenblatt. The perceptron: A probabilstic model for information storage and organisation in the brain. *Phychological Review*, 65:386–408, 1958.

[47] F. Rosenblatt. Two theorems of statistical separability in the perceptron. In *Symposium on Mechanisation of Thought*, volume 1, 1959.

[48] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, USA, 1962.

[49] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, 1:318–362, 1986.

[50] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back–propagating error. *Nature*, 323:533–536, 1986.

[51] J.L. McClelland and D.E. Rumelhart. *Explorations in Parallel Distributed Processing*. MIT press, 1988.

[52] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT press, Cambridge MA, 1972. 2nd edition with corrections (first edition 1969).

[53] R. Callan. *The Essence of Neural Networks*. Prentice Hall, 1999.

[54] Y. Wu, Q. Song, and X. Yang. Robust recurrent neural network control of biped robot. *Journal of Intelligent Robotics Systems*, 49:151–169, June 2007.

[55] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[56] T.G. Barbounis, J.B. Theocharis, M.C. Alexiadis, and P.S. Dokopoulos. Long-term wind speed and power forecasting using local recurrent neural network models. *IEEE Transactions on Energy Conversion*, 21(1):273 – 284, 2006.

[57] J. Vermaak and E.C. Botha. Recurrent neural networks for short-term load forecasting. *IEEE Transactions on Power Systems*, 13(1):126 –132, February 1998.

[58] A.M. Ahmad, S. Ismail, and D.F. Samaon. Recurrent neural network with back-propagation through time for speech recognition. *IEEE International Symposium on Communications and Information Technology (ISCIT 2004)*, 1:98–102, 2004.

[59] H.T. Siegelmann and E.D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4:77–80, 1991.

[60] J. Kilian and H. Siegelmann. The dynamic universality of sigmoidal neural networks. *Information and Computation*, 128:48–56, 1996.

[61] L. Jin, M.M. Gupta, and P.N. Nikiforuk. Universal approximation using dynamic recurrent neural networks: discrete–version. In *IEEE International Conference on Neural Networks (ICNN)*, volume 1, pages 403–408, 1995.

[62] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[63] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[64] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, German National Research Institute for Computer Science, 2001.

[65] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14:2531–2560, 2002.

[66] I. Flood. Next generation artificial neural networks for civil engineering. *Journal of Computing in Civil Engineering*, 20(5):305–307, 2006.

[67] ASCE. Research library. http://ascelibrary.aip.org/, 2006.

[68] J.B. Butcher, P.W. Haycock, and C.R. Day. The application of neural networks in the non destructive testing of structures: A review. Research Institute for the Environment, Physical Sciences & Applied Mathematics, Keele University, Staffordshire, ST5 5BG, UK, 2008.

[69] B.A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM), Keele University, Staffordshire, ST5 5BG, UK, 2007.

[70] D.L. Sackett, S.E. Straus, W.S. Richardson, W. Rosenberg, and R.B. Haynes. *Evidence-Based Medicine How to Practice and Teach EBM.* Churchill Livingstone, Edinburgh, 2000.

[71] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944.

[72] D. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11:431–441, 1963.

[73] D.G. Pratt and J. Lawrence. The use of a neural network in non-destructive testing. California scientific software, 1002 Newtown Rd, Nevada City, CA 95959, 1990.

[74] M. Sansalone and N.J. Carino. Transient impact response of plates containing flaws. *Research of the National Bureau of Standards*, 92(6):117–146, 1987.

[75] S.V. Barai and P.C. Pandey. Vibrational signature analysis using artificial neural networks. *Journal of Computing in Civil Engineering*, 9(4):259–265, 1995.

[76] N. Haritos and J.S. Owen. The use of vibration data for damage detection in bridges: a comparison of system identification and pattern recognition approaches. *Structural Health Monitoring*, 3(2):141–163, 2004.

[77] A. Lorenzi, L.C.P. Silva Filho, and J.L. Campagnolo. Using a back-propagation algorithm to create a neural network for interpreting ultrasonic readings of concrete. In *World Congress on NDT(WCNDT)*, 2004.

[78] Y. Xiang, S.K. Tso, and X.Z. Shi. EIH model and its application on concrete flaw detection. In *15th World Conference on Non Destructive Testing*, 2000.

[79] V. Kanwar, N. Kwatra, and P. Aggarwal. Damage detection for framed RCC buildings using ANN modeling. *International Journal of Damage Mechanics*, 16(4):457–472, 2007.

[80] C.A. Jeyasehar and K. Sumangala. Nondestructive evaluation of prestressed concrete beams using an Artificial Neural Network (ANN) approach. *Structural Health Monitoring*, 5:313–323, 2006.

[81] M.F. Elkordy, K.C. Chang, and G.C. Lee. A structural damage neural network monitoring system. *Microcomputers in Civil Engineering*, 9(2):83–96, 1994.

[82] I. Mikami, S. Tanaka, and T. Hiwatashi. Neural network system for reasoning residual axial forces of high strength bolts in steel bridges. *Computer Aided Civil and Infrastucture Engineering*, 13(4):237–246, 1998.

[83] M.Q. Feng and E.Y. Bahng. Damage assessment of jacketed RC columns using vibrational tests. *Journal of Structural Engineering*, 125(3):265–271, 1999.

[84] Y-Y. Liu, Y-F. Ju, C-D. Duan, and X-F. Zhao. Structure damage diagnosis using neural network and feature fusion. *Engineering Applications of Artificial Intelligence*, 24:87–92, 2011.

[85] J. Cattan and J. Mohammadi. Analysis of bridge condition rating data using neural networks. *Microcomputers in Civil Engineering*, 12:419–429, 1997.

[86] Q. Chen, Y.W. Chan, and K. Worden. Structural fault diagnosis and isolation using neural networks based on response-only data. *Computers and Structures*, 81:2165–2172, 2003.

[87] T. Parthiban, R. Ravi, G.T. Parthiban, S. Srinivasan, K.R. Ramakirshnan, and M. Raghavan. Neural network analysis for corrosion of steel in concrete. *Corrosion Science*, 47:1625–1642, 2005.

[88] V. Jr. Lopes, G. Park, H.H. Cudney, and D.J. Inman. Structural integrity identification based on smart materials and neural networks. In *MAC-XVIII Conference*, volume 1, pages 510–515. SEM, United States, 2000.

[89] M.A. Kewalramani and R. Gupta. Concrete compressive strength prediction using ultrasonic pulse velocity through artificial neural networks. *Automation in Construction*, 15:374–379, 2006.

[90] S. Tapkin, M. Tuncan, Ö. Aröz, A. Tuncan, and K. Ramyar. Estimation of concrete compressive strength by using ultrasonic pulse velocities and artificial neural networks. In *Conference for computer aided engineering and system modelling*, 11th FIGES User's Conference, September 1315, Bolu, Turkey., 2006.

[91] D-S. Yang, S-K. Park, and J-H. Lee. A prediction on mix proportion factor and strength of concrete using neural network. *KSCE Journal of Civil Engineering*, 7(5):525–536, 2003.

[92] R. Gupta, M.A. Kewalramani, and A. Goel. Prediction of concrete strength using neural-expert system. *Journal of Materials in Civil Engineering*, 18(3):462–466, 2006.

[93] T.C.K. Molyneaux, S.G. Millard, J.H. Bungey, and J.Q. Zhou. Radar assessment of structural concrete using neural networks. *NDT&E International*, 28(5):281–288, 1995.

[94] M.R. Shaw, S.G. Millard, T.C.K. Molyneaux, M.J. Taylor, and J.H. Bungey. Location of steel reinforcement in concrete using ground penetrating radar and neural networks. *NDT&E International*, 38(3):203–212, 2005.

[95] M. Zaid, P. Gaydecki, S. Quek, G. Miller, and B. Fernandes. Extracting dimensional information from steel reinforcing bars in concrete using neural networks trained on data from an inductive sensor. *NDT&E International*, 37(7):551–558, 2004.

[96] N.P. de Alcantara Jr and M.E.L. Gasparini. Steel bars identification in reinforced concrete structures by using ANN and magnetic fields. In *Progress in Electromagnetics Research Symposium*, pages 22–26, Hangzhou, China, 2005.

[97] L.H. Yam, Y.J. Yan, and J.S. Jiang. Vibration-based damage detection for composite structures using wavelet transform and neural network identification. *Composite Structures*, 60(4):403–412, 2003.

[98] T. D'Orazio, M. Leo, A. Distante, C. Guaragnella, V. Pianese, and G. Cavaccini. Automatic ultrasonic inspection for internal defect detection in composite materials. *NDT&E International*, 41(2):145–154, 2008.

[99] J.N. Kudva, N. Munir, and P.W. Tan. Damage detection in smart structures using neural networks and finite-element analyses. *Smart materials and structures*, 1(2):108–112, 1992.

[100] A. Kesavan, M. Deivasigamani, S. John, and I. Herszberg. Detection of delaminations in composites structures. In *Proceedings of the International Conference on Artificial Intelligence*, Las Vegas, USA, 2005.

[101] D.S. Hsu and C.H. Tsai. Reinforced concrete structural damage diagnosis by using artficial neural network. In *Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems (IIS '97)*, 1997.

[102] H.C. Das and D.R. Parhi. Application of neural network for fault diagnosis of cracked cantilever beam. In *World Congress on Nature and Biologically Inspired Computing (NABIC)*, pages 1303–1308, 2009.

[103] C.A. Jeyasehar and K. Sumangala. Damage assessment of prestressed concrete beams using artificial neural network (ANN) approach. *Computers and Structures*, 84(26-27):1709–1718, 2006.

[104] M. Mehrjoo, N. Khaji, H. Moharrami, and A. Bahreininejad. Damage detection of truss bridge joints using artificial neural networks. *Expert Systems with Applications*, 35(3):1122–1131, 2008.

[105] X. Wu, J. Ghaboussi, and J.H. Garrett. Use of neural networks in detection of structural damage. *Computers and Structures*, 42(2):649–659, 1992.

[106] D. Maity and A. Saha. Damage assessment in structure from changes in static parameter using neural networks. *Sādhanā*, 29(3):315–327, 2004.

[107] P.M. Pawar, K. Venkatesulu Reddy, and R. Ganguli. Damage Detection in Beams using Spatial Fourier Analysis and Neural Networks. *Journal of Intelligent Material Systems and Structures*, 18(4):347–359, 2007.

[108] P.C. Pandey and S.V. Barai. Multilayer perceptron in damage detection of bridge structures. *Computers & Structures*, 54(4):597–608, 1995.

[109] B. Xu, Z. Wub, G. Chena, and K. Yokoyama. Direct identification of structural parameters from dynamic responses with neural networks. *Engineering Applications of Artificial Intelligence*, 17(8):931–943, 2004.

[110] H. Xu and J. Humar. Damage detection in a girder bridge by artificial neural network technique. *Computer Aided Civil and Infrastucture Engineering*, 21(6):450–646, 2006.

[111] B. Xu, G. Chen, and Z.S. Wu. Parametric identification for a truss structure using axial strain. *Computer Aided Civil and Infrastucture Engineering*, 22(3):210–222, 2007.

[112] Z. Wu, B. Xu, and K. Yokoyama. Decentralized parametric damage detection based on neural networks. *Computer Aided Civil and Infrastucture Engineering*, 17:175–184, 2002.

[113] L. Faravelli and A.A. Pisano. A neural network approach to structure damage assessment. In *Intelligent Information Systems (IIS '97)*, pages 585–588, Grand Bahama Island, Bahamas, 1997.

[114] B. Xu. Neural networks based structural model updating methodology using spatially incomplete accelerations. In J. Licheng, W. Lipo, G. Xinbo, L. Jing, and W. Feng, editors, *Advances in Natural Computation*, volume 4221 of *Lecture Notes in Computer Science*, pages 361–370. Springer Berlin / Heidelberg, 2006.

[115] X. Yuan and S. Gao, C.and Gao. Momentum BP neural networks in structural damage detection based on static displacements and natural frequencies. In *Proceedings of the 4th international symposium on Neural Networks: Advances in Neural Networks, Part III*, ISNN '07, pages 35–40, Berlin, Heidelberg, 2007. Springer-Verlag.

[116] O.R. de Lautour and P. Omenzetter. Damage classification and estimation in experimental structures using time series analysis and pattern recognition. *Mechanical Systems and Signal*, 24(5):1556–1569, 2010.

[117] Z.X. Li and X.M. Yang. Damage identification for beams using ANN based on statistical property of structural responses. *Computers and Structures*, 86(1-2):64–71, 2008.

[118] B. Cannas, S. Carcangiu, F. Cau, A. Fanni, A. Montisci, and P. Testoni. Artificial neural networks for non destructive testing of concrete structures. In *10th Inter-*

national Conference on Engineering Applications of Neural Networks (EANN), 2007.

[119] B. Cannas, S. Carcangiu, F. Cau, A. Fanni, and A. Montisci. Time and frequency approaches to non destructive testing in concrete pillars using neural networks. In *Proceedings of the international conference on Computational Science and Its Applications, Part II*, ICCSA '08, pages 606–616, Berlin, Heidelberg, 2008. Springer-Verlag.

[120] N. Bakhary, H. Hao, and A.J. Deeks. Structure damage detection using neural network with multi-stage substructuring. *Advances in Structural Engineering*, 13(1):1–16, 2010.

[121] J. Hola and Schabowicz. New technique of nondestructive assessment of concrete strength using artificial intelligence. *NDT&E International*, 38(4):251–259, 2005.

[122] I. Haryanto, J. Setiawan, and A. Budiyono. Structural damage detection using randomized trained neural networks. In B. Agus, R. Bambang, and J. Endra, editors, *Intelligent Unmanned Systems: Theory and Applications*, volume 192 of *Studies in Computational Intelligence*, pages 245–255. Springer Berlin / Heidelberg, 2009.

[123] K.V. Yuen and H.F. Lam. On the complexity of artificial neural networks for smart structures monitoring. *Engineering Structures*, 28(7):977–984, 2006.

[124] O. Postolache, H. Geirinhas Ramos, and A. Lopes Ribeiro. Detection and characterisation of defects using GMR probes and artificial neural networks. *Computer Standards & Interfaces*, 33(2):191–200, 2010.

[125] X. Fang, H. Luo, and J. Tang. Structural damage detection using neural network with learning rate improvement. *Computers and Structures*, 83(25-26):2150–2161, 2005.

[126] H. Adeli and J. Zhang. An improved perceptron learning algorithm. *Neural, Parallel, and Scientific Computations*, 1(2):141–152, 1993.

[127] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on neural networks*, pages 586–591, 1993.

[128] L. Ziemiański and G. Harpula. The use of neural networks for damage detection in eight storey frame structure. In *VI Conference on Neural Networks And Their Applications*, pages 332–338, Zakopane, 1999.

[129] B.A. Mogan. Damage detection and localization using probabilistic neural networks. Research Conducted at Keio University, Tokyo, Japan, while attending REUJAT 2005 summer school.

[130] Y.Q. Ni, X.T. Zhou, J.M. Ko, and B.S. Wang. *Advances in Structural Dynamics*, volume 2, chapter Vibration-based damage localization in Ting Kau Bridge using probabilistic neural network, pages 1069–1076. Elsevier Science Ltd, Oxford, UK, 2000.

[131] R. Gil Pita, R.R. Vincen, M. Rosa, M.P. Jarabo, P. Vera, and J. Curpian. Ultrasonic flaw detection using radial basis function networks (RBFNs). *Ultrasonics*, 42(1-9):361–365, 2004.

[132] H.F. Lam, K.V. Yuen, and J.L. Beck. Structural health monitoring via measured ritz vectors utilizing artificial neural networks. *Computer-Aided Civil and Infrastructure Engineering*, 21(4):232–241, 2006.

[133] Y. Yang, J-S. Cheng, G. Ding, and D. Tian. Study on the structural damage indentification method with combined parameters based on RBF neural network. In *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, 2003.

[134] S. Suresh, S.N. Omkar, R. Ganguli, and V. Mani. Identification of crack location and depth in a cantilever beam using a modular neural network approach. *Smart materials and structures*, 13(4):907–915, 2004.

[135] Z. Jianwei and Z. Yina. Damage identification and simulation of structure based on RBF network. In *Mechanic Automation and Control Engineering (MACE), 2010 International Conference on*, pages 1608 –1610, 2010.

[136] A.V. Kappatos and S.E. Dermatas. Feature extraction for crack detection in rain conditions. *Nondestructive Evaluation*, 26(2-4):57–70, 2007.

[137] S. Saadata, M.N. Nooria, G.D. Bucknera, T. Furukawab, and Y. Suzuki. Structural health monitoring and damage detection using an intelligent parameter varying (IPV) technique. *International Journal of Non-Linear Mechanics*, 39(10):1687–1697, 2004.

[138] N. Bakhary, H. Hao, and A.J. Deeks. Damage detection using artificial neural network with consideration of uncertainties. *Engineering Structures*, 29(11):2806–2815, 2007.

[139] L. Mangal, V. G. Idichandy, and C. Ganapathy. ART–based multiple neural networks for monitoring offshore platforms. *Applied Ocean Research*, 18(2-3):137–143, 1996.

[140] H. Adeli. Neural networks in civil engineering: 1989-2000. *Computer Aided Civil and Infrastucture Engineering*, 16(2):126–142, 2001.

[141] T. Kohonen. The self-organising map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[142] T.M. Meksen, B. Boudraa, and M. Boudraa. Defects clustering using kohonen networks during ultrasonic inspection. *International Journal of Computer Science*, 36(3), 2009. Online publication.

[143] M. Wevers, G. Van Dijck, W. Desadeleer, M. Winkelmans, and K. Van Den Abeele. Acoustic emission for on-line monitoring of damage in various application fields. In *European Conference on Acoustic Emission Testing*, pages 587–595, 2004.

[144] Y. Sun. Combined neural network and PCA for complicated damage detection of bridge. In *Fifth International Conference on Natural Computation*, 2009.

[145] S. Roberts and L. Tarassenko. A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6:270–284, 1994.

[146] S.G.A. Thomopoulos, D.K. Bougoulius, and C-D. Wan. DIGNET: An unsupervised learning algorithm for clustering and data fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 31:21–38, 1995.

[147] W.T. Yeung and J.W. Smith. Damage detection in bridges using neural networks for pattern recognition of vibration signatures. *Engineering Structures*, 27:685–698, 2005.

[148] M. Wevers, Van Dijckm G., W. Desadeleer, M. Winkelmans, and K. Van Den Abeele. Acoustic emission for online monitoring of damage in various application fields. In *Proceedings of the 26th European Conference on Acoustic Emission Testing*, pages 587–595, 2004.

[149] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002. ISBN 981-238-151-1.

[150] D.H. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, 1989.

[151] C.M. Tam, T.K.L. Tong, T.C.T. Lau, and K.K. Chan. Diagnosis of prestressed concrete pile defects using probabilistic neural networks. *Engineering Structures*, 26(8):1155–1162, 2004.

[152] O.B. Tokdemir, C. Ayvalik, and J. Mohammidi. Prediction of highway bridge performance by artficial neural networks and genetic algorithms. In *17th International Symposium on Automation and Robotics in Construction*, 2000.

[153] H. Furuta, J. He, and E. Watanabe. A fuzzy expert system for damage assessment using genetic algorithms and neural networks. *Microcomputers in Civil Engineering*, 11(1):37–45, 1996.

[154] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:623–656, 1948.

[155] H. Li, Y. Bao, and J. Ou. Structural damage identification based on integration of information fusion and shannon entropy. *Mechanical Systems and Signal Processing*, 22(6):1427–1440, 2008.

[156] C. Avila, Y. Shiraishi, and Y. Tsuji. Crack width prediction of reinforced concrete structures by artificial neural networks. In *7th Seminar on Neural Network Applications in Electrical Engineering*, 2004.

[157] P. Somervuo and T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2):151–159, 1999.

[158] F.W. Margrave, K. Rigas, D.A. Bradley, and P. Barrowcliffe. The use of neural networks in ultrasonic flaw detection. *Measurement*, 25(2):143–154, 1999.

[159] G. Morcous and Z. Lounis. Prediction of onset of corrosion in concrete bridge decks using neural networks and case-based reasoning. *Computer Aided Civil and Infrastucture Engineering*, 20(2):108–117, 2005.

[160] D. Novák and D. Lehký. ANN inverse analysis based on stochastic small-sample training set simulation. *Engineering Applications of Artificial Intelligence*, 19(7):731–740, 2006.

[161] M.M. Reda Taha and J. Lucero. Damage identification for structural health monitoring using fuzzy pattern recognition. *Engineering Structures*, 27(12):1774–1783, 2005.

[162] E. Altunok, M.M. Reda Taha, D.S. Epp, R.L. Mayes, and T.J. Baca. Damage pattern recognition for structural health monitoring using fuzzy similarity prescription. *Computer Aided Civil and Infrastucture Engineering*, 21(8):549–560, 2006.

[163] M. Cacciola, G. Megali, D. Pellicanò, M. Buonsanti, S. Calcagno, M. Versaci, and F.C. Morabito. Neuro-fuzzy approach for reconstructing fissures in concrete's reinforcing bars. In *Proceedings of the 8th International Workshop on Fuzzy Logic and Applications*, WILF '09, pages 171–178, Berlin, Heidelberg, 2009. Springer-Verlag.

[164] S.F. Jiang, C.M. Zhang, and S. Zhang. Two stage structural damage detection using fuzzy neural networks and data fusion techniques. *Expert Systems with Applications*, 38(1):511–519, 2011.

[165] Y-S. Diao, H-J. Li, and Y. Wang. A two-step structural damage detection approach based on wavelet packet analysis and neural network. In *Fifth International Conference on Machine Learning and Cybernetics*, 2006.

[166] P. Beena and R. Ganguli. Structural damage detection using fuzzy cognitive maps and hebbian learning. *Applied Soft Computing*, 11(1):1014–1020, 2011.

[167] J. Lee, K.F.R. Liu, and W. Chiang. A fuzzy petri net-based expert system and its application to damage assessment of bridges. *IEEE Transactions on Systems, man, and cybernetics – Part B: Cybernetics*, 29(3):350–369, 1999.

[168] Z.W. Lim, C.K-Y. Tan, W.K-G. Seah, and G-H. Tan. Detection of failures in civil structures using artificial neural networks. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, ICANN '09, pages 976–985, Berlin, Heidelberg, 2009. Springer-Verlag.

[169] S.V. Barai and P.C. Pandey. Time-delay neural networks in damage detection of railway bridges. *Advances in Engineering Software*, 28(1):1–10, 1997.

[170] L. Niu. A time-delay neural networks architecture for structural damage detection. In *International Conference on Advanced Computer Control, ICACC '09.*, pages 571 –575, 2009.

[171] C.Y. Kao and S.L. Hung. Detection of structural damage via free vibration responses generated by approximating artificial neural networks. *Computers and Structures*, 81(28-29):2631–2644, 2003.

[172] S.L. Hung and Y.L. Lin. Application of an L-BFGS neural network learning algorithm in engineering analysis and design. In *Proc Second Nat Conf on Struct Engineering*, page 221 230, Nantou, Taiwan, ROC, 1994. Chinese Society of Structural Engineering. (in Chinese).

[173] C-C. Huang and C-H. Loh. Nonlinear identification of dynamic systems using neural networks. *Computer Aided Civil and Infrastucture Engineering*, 16(1):28–41, 2001.

[174] S.L. Hung, C.S. Huang, C.M. Wen, and Y.C. Hsu. Nonparametric identification of a building structure from experimental data using wavelet neural network. *Computer Aided Civil and Infrastucture Engineering*, 18(5):356–368, 2003.

[175] Q. Zhang and A. Benveniste. Wavelet networks. *IEEE Transactions of Neural Networks*, 3(6):889–898, 1992.

[176] X. Jiang and H. Adeli. Dynamic wavelet neural network for nonlinea identification of highrise buildings. *Computer Aided Civil and Infrastucture Engineering*, 20(5):316–330, 2005.

[177] Z. Zhang, G.Z. Liu, and F. Liu. An adaptive wavelet network construction and learning algorithm. *Science in China (Series E)*, 31(2):172–181, 2001.

[178] H. Su, Z. Wu, and Z. Wen. Identification model for dam behavior based on wavelet network. *Computer Aided Civil and Infrastucture Engineering*, 22(6):438–448, 2007.

[179] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In 8, editor, *National Academy of Scientists*, volume 79, pages 2554–2558, 1982.

[180] C.H. Chen. Advanced image processing methods for ultrasonic NDE research. In *16th WCNDT – World Conference on NDT*, 2004.

[181] J.J. Steil. Backpropagation-decorrelation: Recurrent learning with O(N) complexity. In *International Joint Conference on Neural Networks (IJCNN)*, pages 843–848, 2004.

[182] H. Jaeger. A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. Technical Report GMD Report 159, German National Research Center for Information Technology, 2002.

[183] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Neural Information Processing Systems (NIPS)*, 2002.

[184] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[185] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *European Symposium on Artificial Neural Networks (ESANN)*, 2007.

[186] S. Haykin. *Nonlinear Speech Modeling and Applications*, volume 3445 of *Lecture Notes in Computer Science*, chapter Signal Processing in a Nonlinear, NonGaussian, and Nonstationary World, pages 43–53. Springer-Verlag, 2005.

[187] R.M. French. *Encyclopedia of Cognitive Science*, chapter Catastrophic interference in connectionist networks, pages 431–435. Nature Publishing Group, 2003.

[188] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[189] V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[190] M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25(6):821–837, 1964.

[191] J.A.K. Suykens and J. Vandewalle. Recurrent least squares support vector machines. *IEEE Transactions on Circuits and Systems I*, 47(7):1109–1114, 2000.

[192] J. Sun, C. Zheng, Y. Zhou, Y. Bai, and J. Luo. Nonlinear noise reduction of chaotic time series based on multidimensional recurrent LS–SVM. *Neurocomputing*, 71(16-18):3675–3679, 2008.

[193] D. Schneegaß, M. Schaefer, and T. Martinetz. The intrinsic recurrent support vector machine. In *European Symposium on Artficical Neural Networks (ESANN)*, pages 325–330, 2007.

[194] J. Schmidhuber, M. Gagliolo, D. Wierstra, and F. Gomez. Evolino for recurrent support vector machines. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 593–598, 2006.

[195] J. Schmidhuber, D. Wierstra, and M. Gagliolo. Training recurrent networks by evolino. *Neural Computation*, 19(3):757–779, 2007.

[196] G. Holzmann and H. Hauser. Echo state networks with filter neurons and a delay&sum readout. *Neural Networks*, 23(2):244–256, 2010.

[197] W. Maass and C. Bishop. *Pulsed Neural Networks*. Bradford Books/MIT press, 2001.

[198] B. Schrauwen. *Towards applicable spiking neural networks*. PhD thesis, Electronics and Information Systems department, University of Ghent, Belgium, 2008.

[199] W. Maass. *Advances in Neural Information Processing Systems*, chapter Noisy Spiking Neurons with Temporal Coding have more Computational Power than Sigmoidal Neurons, pages 211–217. MIT press, 1997.

[200] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.

[201] L. Büsing, B. Schrauwen, and R. Legenstein. Connectivity, dynamics and memory in reservoir computing with binary and analog neurons. *Neural Computation*, 22(5):1272–1311, 2010.

[202] C. Gallicchio and A. Micheli. Architectural and markovian factors of echo state networks. *Neural Networks*, 24(5):440–456, 2011.

[203] A. Destexhe, M. Rudolph, and D. Paré. The high–conductance state of neocortical neurons in vivo. *Nature Reviews Neuroscience*, 4(9):738–751, 2003.

[204] A.F. Atiya and A.G. Parlos. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, 11(3):697–709, 2000.

[205] E.H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26:394–395, 1920.

[206] R. Penrose. A generalised inverse for matrices. In *Cambridge Philosophical Society*, volume 51, pages 406–413, 1955.

[207] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 4th edition, 2001.

[208] D.C. Montgomery and E.A. Peck. *Introduction to Linear Regression Analysis*. Probability and Mathematical Statistics. Wiley, USA, 1982.

[209] S. Haykin and B. Widrow. *Least-Mean-Square Adaptive Filters*. Wiley, 2003.

[210] D. Verstraeten. *Reservoir Computing: Computation with Dynamical Systems*. PhD thesis, Electronics and Information Systems department, University of Ghent, Belgium, 2009.

[211] F. Wyffels and B. Schrauwen. A comparative study of Reservoir Computing strategies for monthly time series prediction. *Neurocomputing*, 73(10-12):1958–1964, 2010.

[212] J. Triesch. *Advances in Neural Information Processing Systems*, volume 17, chapter Synergies between Intrinsic and Synaptic Plasticity in Individual Model Neurons, pages 1417–1424. MIT Press, Cambridge, MA, 2005.

[213] R. Baddeley, L.F. Abbott, M.C.A. Booth, F. Sengpeil, T. Freeman, E.A. Wakeman, and Rolls. E.T. Responses of neurons in primary and inferior temporal

visual cortices to natural scenes. In *Proceedings of the Royal Society of London*, volume B 264, pages 1775–1783, 1997.

[214] J.J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Network*, 20(3):353–364, 2007.

[215] M. Wardermann and J. Steil. Intrinsic plasticity for reservoir learning algorithms. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 513–518, 2007.

[216] B. Schrauwen, M. Wardermann, D. Verstraeten, J.J. Steil, and D. Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159–1171, 2008.

[217] D. Verstraeten, Schrauwen B., and Stroobandt D. Adapting reservoirs to get gaussian distributions. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 495–500, 2007.

[218] A. Rodan and P. Tiňo. Simple deterministically constructed recurrent neural networks. In C. Fyfe, P. Tiňo, D. Charles, C. Garcia-Osorio, and H. Yin, editors, *Intelligent Data Engineering and Automated Learning (IDEAL)*, volume 6283 of *Lecture Notes in Computer Science*, pages 267–274. Springer Berlin / Heidelberg, 2010.

[219] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert. Optimisation and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

[220] K Ishii, T. van der Zant, V. Becanovic, and P. Ploger. Identification of motion with echo state network. In *OCEANS '04. MTTS/IEEE TECHNO-OCEAN '04*, pages 1205–1210, 2004.

[221] F. Jiang, H. Berry, and M. Schoenauer. Supervised and evolutionary learning of echo state networks. In *International Conference on Parallel Problem Solving From Nature*, 2008.

[222] U.D. Schiller and J.J. Steil. Analyzing the weight dynamics of recurrent learning algorithms. *Neurocomputing*, 63:5–23, 2005.

[223] R.J. Williams and D. Zisper. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[224] W. Maass, T. Natschläger, and H. Markram. *Advances in Neural Information Processing Systems*, volume 15, chapter A model for real-time computation in general neural microcircuits, pages 213–220. MIT press, 2003.

[225] W. Maass, P. Joshi, and E.D. Sontag. *Advances in Neural Information Processing Systems*, volume 18, chapter Principles of real-time computing with feedback applied to cortical microcircuit models, pages 835–842. MIT press, 2006.

[226] D. Verstraeten, J. Dambre, X. Dutoit, and B. Schrauwen. Memory versus non-linearity in reservoirs. In *Internation Joint Conference on Neural Networks (IJCNN)*, pages 2669–2676, 2010.

[227] X. Dutoit. *Reservoir Computing for Intelligent Mobile Systems*. PhD thesis, Department of Mechanical Engineering, Katholleke Universiteit Leuven, Belgium, 2009.

[228] B. Schrauwen, J. Defour, D. Verstraeten, and J. Van Campenhout. The introduction of time-scales in reservoir computing. In *International Conference on Artificial Neural Networks (ICANN)*, pages 471–479, 2007.

[229] H. Jaeger. Short term memory in echo state networks. Technical report, German National Research Centre for Information Technology, 2002.

[230] F. Wyffels, B. Schrauwen, and D. Stroobandt. Stable output feedback in reservoir computing using ridge regression. In *International Conference on Artificial Neural Networks*, pages 808–817, 2008.

[231] B. Noris, M. Nobile, L. Piccinini, M. Berti, E. Mani, M. Molteni, F. Keller, D. Campolo, and A.G. Billard. Gait analysis of autistic children with echo-state networks. In *Neural Information Processing Systems (NIPS)*, 2006.

[232] S. Jarvis, S. Rotter, and U. Egert. Extending stability through hierarchical clusters in Echo State Networks. *Frontiers in Neuroinformatics*, 4(11), 2010.

[233] D. Verstraeten, B. Schrauwen, and D. Stroobandt. Reservoir-based techniques for speech recognition. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1050–1053, 2006.

[234] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006.

[235] M.C. Ozturk, D. Xu, and J.C. Principe. Analysis and design of echo state networks. *Neural Computation*, 19(1):111–138, 2007.

[236] N. Bertschinger and T. Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004.

[237] R. Legenstein and W. Maass. *New directions in statistical signal processing: From systems to brain*, chapter What makes a dynamical system computationally powerful? MIT press, 2005.

[238] R. Legenstein and W. Maass. 2007 special issue: Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007.

[239] M.C. Ozturk and J.C. Principe. Computing with transiently stable states. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1467–1472, 2005.

[240] D. Verstraeten and B. Schrauwen. On the quantification of dynamics in reservoir computing. In *International Conference on Artificial Neural Networks (ICANN)*, pages 985–994, 2009.

[241] K. Alligood, T. Sauer, and J. Yorke. *Chaos: An Introduction to Dynamical Systems*. Springer, 1996.

[242] A. Renyi. *Probability Theory*. Dover Publications, 1970.

[243] M. Hermans and B. Schrauwen. Memory in reservoirs for high dimensional input. In *Internation Joint Conference on Neural Networks (IJCNN)*, pages 2662–2668, 2010.

[244] L. Feldkamp, D. Prokhorov, F. Eagen, and F. Yuan. *Nonlinear modelling: Advanced Black-Box techniques*, chapter Enhanced multi-stream Kalman filter training for recurrent neural networks, pages 29–54. Kluwer, 1998.

[245] J. Vesanto. Using the SOM and local models in time-series prediction. In *Workshop on Self-Organising Maps (WSOM)*, pages 209–214, 1997.

[246] D.F. Findley, B.C. Mosell, M.C. Bell, W.R. Otto, and B.C. Chen. New capabilities and methods of the X–12–ARIMA seasonal–adjustment program. *Journal of Business and Economic Statistics*, 16(2):127–152, 1996.

[247] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with a liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.

[248] G. Doddington and T. Schalk. Speech recognition: Turning theory to practice. *IEEE Spectrum*, 18(9):26–32, 1981.

[249] B. Schrauwen and L. Büsing. A hierarchy of recurrent networks for speech recognition. In *Neural Information Processing Systems (NIPS)*, 2009.

[250] E.N. Lorenz. *The Essence of Chaos*. University of Washington Press, Seattle, WA, 1995.

[251] M. Hermans and B. Schrauwen. One step backpropagation through time for learning input mapping in reservoir computing applied to speech recognition. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 521 –524, June 2010.

[252] M.D. Skowronski and J.G. Harris. Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, 20(3):414–423, 2007.

[253] M.D. Skowronski and J.G. Harris. Noise robust automatic speech recognition using a discriminative echo state network. In *Internation Symposium on Circuits and Systems*, pages 1771 – 1774, 2007.

[254] B.H. Juang and L.R. Rabiner. The segmental K–means algorithm for estimating parameters for hidden Markov models. *Institute of Electrical and Electronics*

Engineers Transactions on Acoustics, Speech and Signal Processing, 38(9):1639–1641, 1990.

[255] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J-P. Martens. Phoneme recognition with large hierarchical reservoirs. In *Neural Information Processing Systems (NIPS)*, 2010.

[256] G. Holzmann. Reservoir computing: A powerful black-box framework for nonlinear audio processing. In *12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, September 1-4 2009.

[257] E. Antonelo, B. Schrauwen, and D. Stroobandt. Mobile robot control in the road sign problem using reservoir computing networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 911–916, 2008.

[258] J. Hertzberg, H. Jaeger, and F. Schonherr. Learning to ground fact symbols into behaviour-based robots. In *15th European Conference on Artificial Intelligence*, volume 77, pages 708–712, 2002.

[259] M. Salmen and P.G. Plöger. Echo state networks used for motor control. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1953 – 1958, 2005.

[260] P.G. Plöger, A. Arghir, T. Günther, and R. Hosseiny. *RoboCup 2003: Robot Soccer World Cup VII*, chapter Echo State Networks for Mobile Robot Modeling and Control, pages 157–168. Springer Berlin / Heidelberg, 2004.

[261] E. Antonelo, B. Schrauwen, X. Dutoit, D. Stroobandt, and M. Nuttin. Event detection and localization in mobile robot navigation using reservoir comput-

ing. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 660–669, Berlin, Heidelberg, 2007. Springer-Verlag.

[262] E. Antonelo, B. Schrauwen, and D. Stroobandt. Modeling multiple autonomous robot behaviors and behavior switching with a single reservoir computing network. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1843–1848, 2008.

[263] E. Antonelo, B. Schrauwen, and D. Stroobandt. Imitation learning of an intelligent navigation system for mobile robots using reservoir computing. In *Proceedings of the 10th Brazilian Symposium on Neural Networks (SBRN 2008)*, pages 93–98, 2008.

[264] E. Antonelo and B. Schrauwen. *Artificial Neural Networks ICANN 2009*, volume 5768 of *Lecture Notes in Computer Science*, chapter Unsupervised learning in reservoir computing: modeling hippocampal place cells for small mobile robots, pages 747–756. Springer Berlin / Heidelberg, 2009.

[265] T. Waegemann, E. Antonelo, F. Wyffels, and B. Schrauwen. Modular reservoir computing networks for imitation learning of multiple robot behaviours. In *8th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 27–32, 2009.

[266] C. Hartland, N. Bredeche, and M. Sebag. Memory-enhanced evolutionary robotics: the echo state network approach. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, CEC'09, pages 2788–2795, Piscataway, NJ, USA, 2009. IEEE Press.

[267] E. Antonelo and B. Schrauwen. Supervised learning of internal models for autonomous goal-oriented robot navigation using reservoir computing. In *IEEE International conference on Robotics and Automation*, pages 2959–2964, 2010.

[268] O. Obst, X.R. Wang, and M. Prokopenko. Using echo state networks for anomaly detection in underground coal mines. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 219–229. IEEE Computer Society, 2008.

[269] M. Chang, A. Terzis, and P. Bonnet. Mote-based online anomaly detection using echo state networks. In *5th IEEE International Conference on Distributed Computing in Sensor Systems*, volume 5516 of *Lecture Notes in Computer Science*, pages 72–86, 2009.

[270] O. Obst. Distributed fault detection using a recurrent neural network. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 373–374, 2009.

[271] A. Devert, N. Bredeche, and M. Schoenauer. *Artificial Evolution*, volume 4926 of *Lecture Notes in Computer Science*, chapter Unsupervised learning of Echo State Networks: A Case Study in Artificial Embryogeny, pages 278–290. Springer, Berlin/Heidelberg, 2008.

[272] K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[273] S.L. Frank and H. Jacobsson. Sentence processing in echo state networks: a qualitative analysis by finite state machine extraction. *Connection Science*, 22(2):135–155, 2010.

[274] M.H. Tong, A.D. Bickett, E.M. Christiansen, and G.W. Cottrell. Learning grammatical structure with echo state networks. *Neural Networks*, 20(3):424–432, 2007.

[275] M. Čerňanský and P. Tiňo. *Artificial Neural Networks - ICANN*, chapter Comparison of echo state networks with simple recurrent networks and variable length markov models on symbolic sequences, pages 618–627. Springer, 2007.

[276] E. Frank. Learn more by training less: Systematicity in sentence processing by recurrent networks. *Connection Science*, 18(3):287–302, 2006.

[277] Y.N. Rao, S.P. Kim, J.C. Sanchez, D. Erdogmus, J.C. Principe, J.M. Carmena, M.A. Lebedev, and M.A. Nicolelis. Learning mappings in brain machine interfaces with echo state networks. In *IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 233–236, 2005.

[278] G.K. Venayagamoorthy. Online design of an echo state network based wide area monitor for a multimachine power system. *Neural Networks*, 20(3):404–413, 2007.

[279] A. Gunduz, M.C. Ozturk, J.C. Sanchez, and J.C. Principe. Echo state networks for motor control of human ecog neuroprosthetics. In *3rd International IEEE/EMBS Conference on Neural Engineering*, pages 514 – 517, 2007.

[280] R.B.H.S. Gowrishankar and P.S. Satyanarayana. Neural network based ber prediction for 802.16e channel. In *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5, 2007.

[281] C. Gallicchio and A. Micheli. Graph echo state networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2159–2166, 2010.

[282] H. Showkati, A.H. Hejazi, and S. Elyasi. Short term load forecasting using echo state networks. In *International Joint Conference on Neural Networks (IJCNN)*, 2010.

[283] P. Buteneers, B. Schrauwen, D. Verstraeten, and D. Stroobandt. *Advances in Neuro-Information Processing*, volume 5506 of *Lecture Notes in Computer Science*, chapter Real-time Epileptic Seizure Detection on Intra-cranial Rat Data using Reservoir Computing, pages 56–63. Springer Berlin / Heidelberg, 2008.

[284] P. Buteneers, D. Verstraeten, P. van Mierlo, T. Wuykhuys, S. Staelens, D. Stroobandt, and B. Schrauwen. Automatic detection of epileptic seizures on intra-cranial EEG from rats using reservoir computing. *AI in Medicine*, In press, 2011.

[285] J.B. Butcher, D. Verstraeten, B. Schrauwen, C.R. Day, and P.W. Haycock. Extending reservoir computing with random static projections: a hybrid between OP-ELM and RC. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 303–308, 2010.

[286] J.B. Butcher, D. Verstraeten, B. Schrauwen, C.R. Day, and P.W. Haycock. Pruning reservoirs with random static projections. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 250–255, 2010.

[287] G.B. Huang, Q.Y. Zhu, and C.H. Siew. Extreme learning machines: Theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[288] Y. Miche, A. Sorajamaa, and A. Lendasse. OP-ELM: Theory, experiments and a toolbox. In V. Kurkova, R. Neruda, and J. Koutnik, editors, *Artificial Neural*

Networks - ICANN 2008, LNCS 5163, pages 145–154, Berlin / Heidelberg, 2008. Springer.

[289] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[290] T. Similä and J. Tikka. *International Conference on Artificial Neural Networks*, chapter Multiresponse sparse regression with application to mulidimensional scaling, pages 97–102. Springer, 2005.

[291] G.R. Feng, G.B. Huang, Q.P. Lin, and R. Gay. Error minimised extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20(8):1352–1357, 2009.

[292] Y. Lan, Y.C. Soh, and G.B. Huang. Random search enhancement of error minimised extreme learning machine. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 327–332, 2010.

[293] G.B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16-18):3056–3062, 2007.

[294] Q.Y. Zhu, A.K. Qin, P.N. Suganthan, and G.B. Huang. Evolutionary extreme learning machine. *Pattern Recognition*, 38(10):1759–1763, 2005.

[295] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimisation over continuous spaces. *Global Optimisation*, 11(4):341–359, 1997.

[296] Q. Liu, Q. He, and Z. Shi. Extreme support vector machine classifier. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 222–233. Springer-Verlag, Berlin, Heidelberg, 2008.

[297] B. Frénay and M. Verleysen. Using SVMs with randomised feature spaces: an extreme learning approach. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 315–320, 2010.

[298] G.B. Huang, X. Ding, and H. Zhou. Optimisation method based extreme learning machine for classification. *Neurocomputing*, 74(1-3):155–163, 2010.

[299] H.J. Rong, G.B. Huang, N. Sundarajan, and P. Saratchandran. Online sequential fuzzy extreme learning machine for function approximation and classification problems. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 39(4):1067–1072, 2009.

[300] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions of Neural Networks*, 1(1):4–27, 1990.

[301] W-Y. Deng, Q-H. Zheng, S. Lian, L. Chen, and X. Wang. Ordinal extreme learning machine. *Neurocomputing*, 74(1-3):447–456, 2010.

[302] N.Y. Liang, P. Saratchandran, G.B. Huang, and N. Sundararajan. Classification of mental tasks from EEG signals using extreme learning machine. *International Journal of Neural Systems*, 16(1):29–38, 2006.

[303] R. Minhas, A.A. Mohammed, and Q.M.J. Wu. A fast recognition framework based on extreme learning machine using hybrid object information. *Neurocomputing*, 73(10-12):1831–1839, 2010.

[304] H.J. Rong, G.B. Huang, and Y.S. Ong. Extreme learning machine for multi-categories classification applications. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1709–1713, 2008.

[305] R. Zhang, G.B. Huang, N. Sundarajan, and P. Saratchandran. Multicategory classification using an ELM for gene expression for cancer diagnosis. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(3):485–495, 2007.

[306] Q.J. Benedict and S. Emmanuel. ELM for classification of music genres. In *9th International Conference on Control, Automation, Robotics and Vision, 2006. ICARCV '06.*, pages 1–6, 2006.

[307] M. van Heeswijk, Y. Miche, T. Lindh-Knuutila, P.A.J. Hilbers, T. Honkela, E. Oja, and A. Lendasse. Adaptive ensemble models of extreme learning machines for time series prediction. In *Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN)*, pages 305–314. Springer-Verlag, Berlin, Heidelberg, 2009.

[308] A. Weigand and N. Gershenfeld. *Time series prediction: Forecasting the Future and Understanding the past*. Addison-Wesley, 1994.

[309] R. Singh and S. Balasundaram. Application of extreme learning machine method for time series analysis. *International Journal of Intelligent Technology*, 2(4):256–262, 2007.

[310] F.L. Chen and T.Y. Ou. Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry. *Expert Systems with Applications*, 38(3):1336–1345, 2011.

[311] B. Li, Y. Li, and X. Rong. Comparison of echo state network and extreme learning machine on nonlinear prediction. *Journal of Computational Information Systems*, 7(6):1863–1870, 2011.

[312] X. Dutoit, H. Van Brussel, and M Nuttin. A first attempt of reservoir pruning for classification problems. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 507–512, 2007.

[313] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M Nuttin. Pruning and regularization in reservoir computing: a first insight. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 1–6, 2008.

[314] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin. Pruning and regularization in reservoir computing. *Neurocomputing*, 72(7-9):1534–1546, 2009.

[315] Y. Miche, P. Bas, O. Jutten, C. Simula, and A. Lendasse. OP-ELM Matlab toolbox. Technical report, Adaptive Informatics Research Centre, Department of Information and Computer Science, Helsinki University of Technology, Available for free from `http://www.cis.hut.fi/projects/tsp/index.php?page=research&subpage=opelm`, 2008.

[316] B. Schrauwen, D. Verstraeten, M. D'Haene, and M. Wardermann. Reservoir computing toolbox manual. Available from: `http://reslab.elis.ugent.be/rctoolbox`.

[317] J.L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[318] A. Waibel, T. Hanazawa, G. Hilton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[319] P.J. Werbos. *Beyond regression: new tools for prediction and analysis in the behavioural sciences*. PhD thesis, Harvard University, 1974.

[320] M. Čerňanský. Recurrent neural network simulator. Available from `http://www2.fiit.stuba.sk/~cernans/main/download.html`.

[321] R.J. Williams and D. Zisper. *Backpropagation: Theory, Architectures and Applications*, chapter Gradient-based learning algorithms for recurrent networks and their computational complexity, pages 433–486. Hillsdale, NJ, 1995.

[322] B. Schrauwen, L. Büsing, and R. Legenstein. On computational power and the order-chaos phase transition in reservoir computing. In *Neural Information Processing Systems Conference*, pages 1425–1432, 2008.

[323] R. Lyon. A computational model of filtering, detection and compression in the cochlea. In *IEEE ICASSP*, pages 1282–1285, 1982.

[324] S. Seneff. A joint synchrony/mean–rate model of auditory speech processing. *Jounral of Phonetics*, 16(1):101–111, 1988.

[325] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[326] A.V. Benin, A.S. Semenov, and S.G. Semenov. Modelling of fracture process in concrete reinforced structures under steel corrosion. *Journal of Achievements in Materials and Manufacturing Engineering*, 39(2):168–175, 2010.

[327] L.T.N. Dao, V.T.N. Dao, S.H. Kim, and K.Y. Ann. Modeling steel corrosion in concrete structures - part 2: A unified adaptive finite element model for

simulation of steel corrosion. *International Journal of Electrochemical Science*, 5(3):314–326, 2010.

[328] S. C. Kranc and Alberto A. Sags. Detailed modeling of corrosion macrocells on steel reinforcing in concrete. *Corrosion Science*, 43(7):1355 – 1372, 2001.

[329] G. Balabanić, N. Bićanić, and A. Dureković. The influence of w/c ratio, concrete cover thickness and degree of water saturation on the corrosion rate of reinforcing steel in concrete. *Cement and Concrete Research*, 26(5):761 – 769, 1996.

[330] S.L. Sherratt. Doctoral progression report. Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM), Keele University, Staffordshire, ST5 5BG, UK, 2007.

[331] A.U. Gehring and A.M. Hofmeister. The transformation of lepidocrocite during heating: a magnetic and spectroscopic study. *Clays and Clay materials*, 42(4):409–415, 1994.

[332] H.G. Wheat. Long-term investigation of alternative reinforcement materials for concrete. In M. Grantham, C. Majorana, and V. Salomoni, editors, *Concrete Solutions*, pages 227–231. CRC Press, 2009.

[333] ASTM G109. Standard test method for determining effects of chemical admixtures on corrosion of embedded steel reinforcement in concrete exposed to chloride environments. Technical report, ASTM International.

[334] D.G. Rees. *Essential Statistics*. Chapman and Hall, London, UK, 4th edition, 2000.

[335] S. Hochreiter and J. Schmidhuber. Long Short–Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[336] F. Gers. *Long Short-Term Memory in Recurrent Neural Networks.* PhD thesis, Présentée au Départment d'Informatique, École Polytechnique Fédérale de Lausanne, 2001.

[337] A. Graves, D. Eck, N. Beringer, and J. Schmidhuber. Biologically Plausible speech recognition with LSTM neural nets. In *Bio-ADIT*, pages 127–136, 2004.

[338] P. Lamere, P. Kwok, W. Walker, E. Gouvea, R. Singh, B. Raj, and P. Wolf. Design of the CMU Sphinx–4 decoder. In *Proceedings of the 8th European Conference on Speech Communication and Technology*, pages 1181–1184, 2003.

[339] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx–4: A flexible open source framework for speech recognition. Technical report, Sun Microsystems Inc., 2004.

[340] D. Verstraeten. Een studie nan de liquid state machine: een woordherkenner. Master's thesis, Electronics and Information Systems department, University of Ghent, Belgium, 2004.

[341] B.W. Matthews. Comparison of the predicted and observed secondary structure of T4 phage Iysozyme. *Biochem. Biophys. Acta*, 405:442–451, 1975.

[342] P. Baldi, S. Brunak, Y. Chauvin, C.A.F. Andersen, and H. Neilsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics review*, 16(5):412–424, 2000.

[343] M. Hermans and B. Schrauwen. Time window regularizing using marginal likelihood gradients. Submitted to NIPS 2011, 2011.

[344] R.M. Cornell and U. Schwertmann. *The iron oxides: structure, properties, reactions, occurencies and uses.* Wiley-VCH GmbH & Co, 2nd edition, 2003.

[345] G.S. Duffó, W. Morris, I. Raspini, and C. Saragovi. A study of steel rebars embedded in concrete during 65 years. *Corrosion Science*, 46(9):2143–2157, 2004.

[346] J.J. Santana Rodríguez, F.J. Santana Hernández, and J.E. González Gonázlez. XRD and SEM studies of the layer of corrosion products for carbon steel in various different environments in the province of Las Palmas (The Canary Islands, Spain). *Corrosion Science*, 44(11):2425–2438, November 2002.

[347] R. Vera, M. Villarroel, A.M. Carvajal, E. Vera, and C. Ortiz. Corrosion products of reinforcement in concrete in marine and industrial environments. *Materials Chemistry and Physics*, 114(1):467–474, 2009.

[348] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *The Journal of Systems and Software*, 80(4):571–583, 2007.

[349] J. Bailey, D. Budgen, M. Turner, B.A. Kitchenham, P. Brereton, and S. Linkman. Evidence relating to object orientated software design: A survey. In *1st International Symposium on Empirical Software Engineering and Management*, pages 482–484, Madrid, Spain, September 2007.

[350] P. Woodall and P. Brereton. A Systematic Literature Review from the Perspective of a PhD Student. Technical report, Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM), Keele University, Staffordshire, ST5 5BG, UK, 2006.

[351] D. Budgen, B.A. Kitchenham, P. Brereton, M. Turner, S. Charters, and S. Linkman. Observations arising from performing a systematic review of the technology acceptance model. Technical report, Department of Computer Sci-

ence, Durham University, UK and School of Computing and Mathematics, Keele University, UK, 2007.

[352] J.B. Butcher, P.W. Haycock, and C.R. Day. Protocol for performing a systematic mapping exercise for the application of neural networks in detecting damage in structures. 2008.

[353] B.A. Kitchenham, P. Brereton, S. Owen, J. Butcher, and C. Jefferies. Length and readability of structured software engineering abstracts. *IET Software*, 2(1):37–45, 2008.

[354] ASTM C192. Standard practice for making and curing concrete test specimens in the laboratory. Technical report, ASTM International.

Appendix A

The corrosion process

A.1 Introduction

Corrosion of steel within reinforced concrete occurs mainly as a result of either chloride based attacks, such as road salt and marine environments, or as a result of carbonation. After the degradation of the passive layer, two electrochemical reactions occur creating an anodic and cathodic reaction as explained by Broomfield [10]. The anodic reaction involves the emission of electrons from the steel which occurs as a result of the corrosion of the reinforcing steel inside the concrete:

$$Fe \rightarrow Fe^{2+} + 2e^{-} \tag{A.1}$$

These free electrons must be consumed by some other process. In the presence of water and oxygen the free electrons are consumed in a cathodic reaction to give hydroxyl ions:

$$2e^{-} + H_2O + \frac{1}{2}O_2 \rightarrow 2OH^{-} \tag{A.2}$$

The Fe^{2+} created in equation A.1 is soluble, so no cracking or spalling occurs as the rust dissolves in the water. In order for rust to form, further processes must also occur. One possibility is that the ferrous hydroxide created from the reaction between the ferrous irons and hydroxyl groups (shown in equation A.3), turns into ferric hydroxide (as shown in equation A.4) and eventually hydrated ferric oxide which is rust (as shown in equation A.5). This process is also shown in Figure A.1.

$$Fe^{2+} + 2OH^{-} \rightarrow Fe(OH)_2 \tag{A.3}$$

$$4Fe(OH)_2 + O_2 + 2H_2O \rightarrow 4Fe(OH)_3 \tag{A.4}$$

$$2Fe(OH)_3 \rightarrow Fe_2O_3.H_2O + 2H_2O \tag{A.5}$$

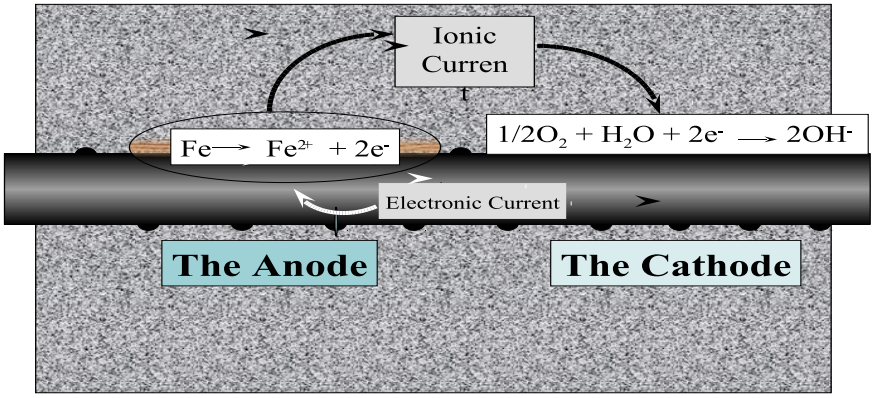


Figure A.1: The various corrosion reactions on steel. Firstly the reaction at the anode strips electrons from the iron which are then consumed at the cathode to form hydroxyl ions. The combination of these two products results in the formation of ferrous hydroxide. In the presence of water and oxygen this then turns into ferric hydroxide and finally hydrated ferric oxide, or rust (kindly supplied by John Broomfield).

A.2 Carbonation

Carbonation is a process which involves carbon dioxide from the air dissolving in the pore water of the cement (shown in equation A.6), which over time has the effect of neutralising the alkalis in the cement as the aqueous carbon dioxide solution creates a weak acid. The process of carbonation inside reinforced concrete can be summarised as follows:

$$CO_2 + H_2O \rightarrow H_2CO_3 \tag{A.6}$$

$$H_2CO_3 + Ca(OH)_2 \rightarrow CaCO_3 + 2H_20 \tag{A.7}$$

Equation A.7 shows the reaction between the carbonic acid and the calcium hydroxide in the concrete, which forms calcium carbonate and water. Carbonation at the steel surface is mainly caused by an insufficient amount of concrete cover over the steel which eases the passage of carbon dioxide to the steel via the pores of the concrete. However, it can also occur at high cover depths when the concrete is very porous which is usually a result of poor concrete manufacture via low cement content (this results in a low amount of alkalis present in the concrete, which helps to speed up the neutralisation of the concrete) and a high water:cement ratio. Poor curing (soaking of the concrete in water during the setting period) can also create porous concrete structures which are more susceptible to carbonation as it leads to open pores which are more likely to be connected to each other [10]. In the built environment, a deep carbonation front is, therefore, often found in structures which have been poorly constructed and that possess some or all of the characteristics stated above. Carbonation can also occur in structures that are in a marine environment as the constant cycle of wetting and drying caused by tidal forces aids the process. During the dry period, carbon dioxide penetrates the concrete and is then dissolved during the wet cycle giving carbonic acid, as shown in equation A.6. Figure A.2 shows the carbonation process in relation to the pH level of the concrete and the concrete cover depth. Carbonation of the rebars can be described as corrosion on a microcell level due to the fact that the corrosion is often found to be at a similar level within close proximity across the rebars of a structure which is often not the case in the chloride induced corrosion presented later.

A.3 Chloride attack

Chlorides are very aggressive agents which attack the protective passive layer formed between the steel and the concrete, allowing corrosion of the rebars to take place.

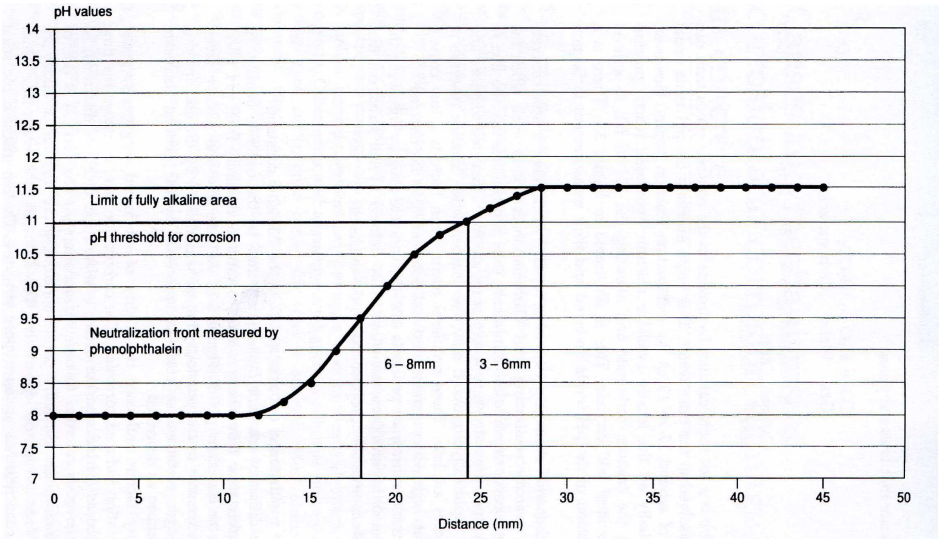


Figure A.2: The relationship between the carbonation process, the concrete cover depth (X axis) and the pH level of the concrete (Y axis). The phenolphthalein measurement shown is used as an indicative method to indicate whether carbonation has occurred, as shown by the neutralisation front (reproduced from Broomfield [10]).

Unlike carbonation, chlorides do not reduce the overall pH level. The breakdown of the passive layer occurs once a chloride concentration threshold has been exceeded which is roughly 0.4% chloride by weight of cement if chlorides are added to the cement during creation or 0.2% through diffusion [10]. This threshold depends on many factors such as environmental conditions and the quality of the concrete. For example, if the quality of the cement is poor, these concentration levels can be reduced, whereas in the the opposite case, chloride concentrations of 1% have sometimes led to no corrosion being observed.

In structures where chloride induced corrosion is present, a macrocell phenomenon is often observed. This occurs when areas of chloride induced corrosion are separated by healthy, clean rebars. This forms as a result of high water concentrations within the concrete (for chlorides to penetrate the concrete they require transportation by water) which in turn increases the electrical conductivity of the concrete. Here, the anode and cathode (described earlier in this section) present in the corrosion process can be further apart as the ions can move further through the concrete. This can be problematic for corrosion detection techniques which are very localised: because the probability of detecting areas of chloride induced corrosion can be reduced depending on the size and location of the areas chosen by the engineer for testing.

The main sources of chlorides are found in marine environments where structures are close to the sea and in transportation where de-icing salts are used to keep roads ice-free during the winter months. Up until the 1970's calcium chloride was added to the concrete to aid setting until it was found to be a cause of corrosion. The use of sea water and/or materials extracted from the sea also introduces chlorides into the concrete. As the corrosion process is better known nowadays, these approaches are

avoided in modern building practices.

A.4 Products of corrosion

The products created as a result of the corrosion process can vary depending on the condition of the concrete and the environment surrounding it. Table A.1 shows the different oxides formed under different conditions.

Table A.1: The different products formed as a result of corrosion under different conditions (reproduced from Cornell and Schwertmann [344]).

Type of corrosion	**Conditions**	**Products formed**
Electrochemical	Stagnant pure water with sufficient oxygen (often found in pipes)	Geothite and lepidocrocite
	Boiling water low in oxygen and/or acid (often found in pipes)	Magnetite
	Hot oxygenated water (often found in pipes)	Magnetite, lepidocrocite and green rust
	Seawater	Magnetite, Lepidocrocite, goethite and akaganeite
Atmospheric	Temperate and tropical environments	Lepidocrocite, goethite and magnetite
	High SO_2	Mainly goethite
	High CL^-	Mainly akageneite
Passive layer	Anodic polarisation in $KOH/NaOH$ and H_2SO_4	Magnetite and hematite
	Borate buffer with Fe^{2+}	Lepidocrocite
	Concentrated HNO_3	Spinel
Thermal	Air at room temperature	Magnetite and maghemite
	Air at 250-550°C	Magnetite and hematite
	Air at 600°C	Wüstite, magnetite and hematite

Although all of the major iron oxides have been observed as products of iron and steel corrosion, typical corrosion products include lepidocrocite ($\gamma - FeOOH$), goethite ($\alpha - FeOOH$), magnetite (Fe_3O4), maghemite ($\gamma - Fe_2O_3$) and hematite ($\alpha - Fe_2O_3$) [344, 345]. In certain conditions, for example marine environments, akaganeite ($\beta - FeOOH$) can also be found [344, 346, 347]. The formation and transformations of the major iron oxides are shown in Figure A.3.

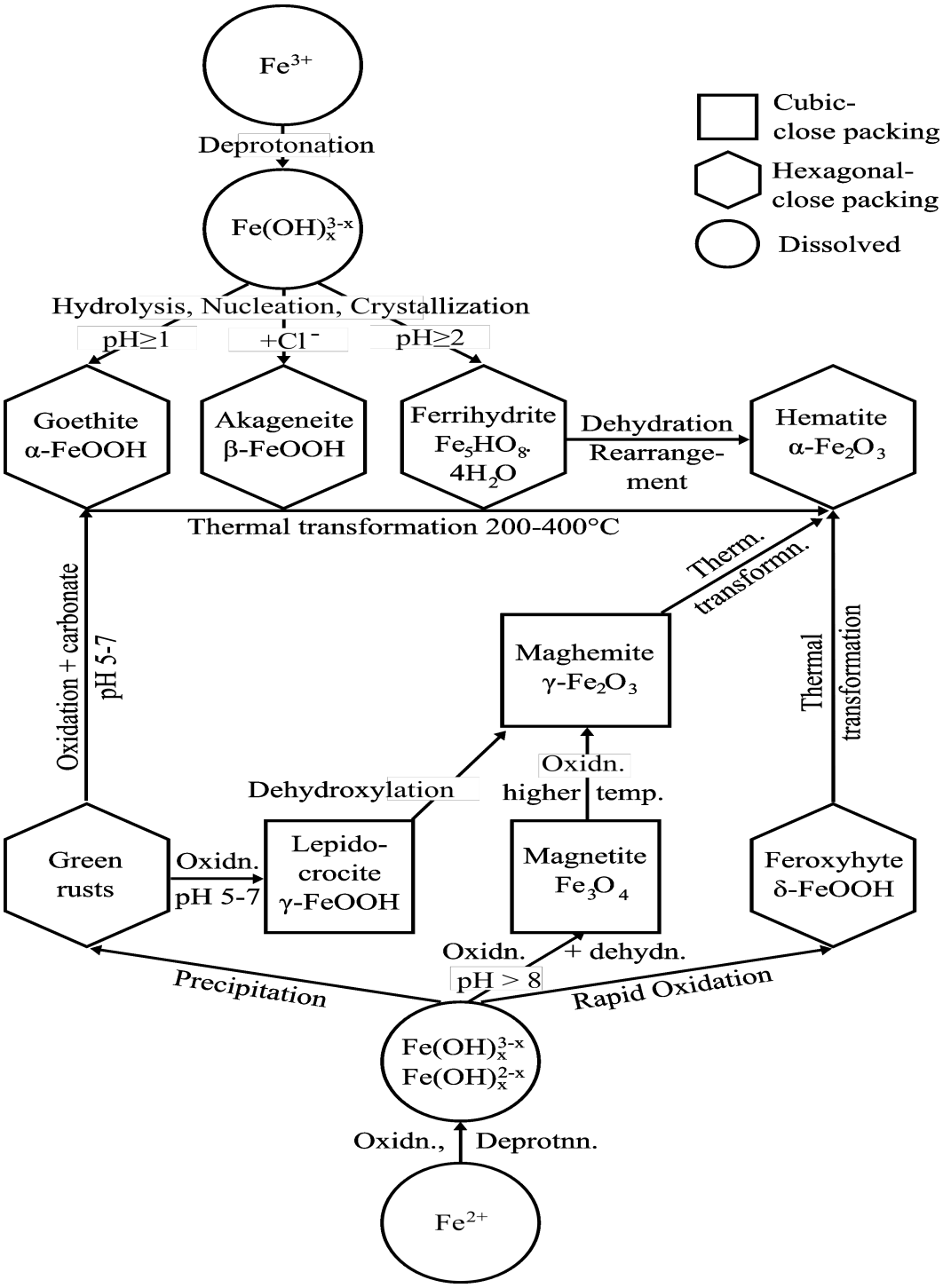


Figure A.3: An overview of the different iron oxide formations and their transformation processes (reproduced from Cornell and Schwertmann [344]).

Appendix B

The application of Neural Networks in the Non Destructive Testing of Structures: A Review

Abstract

Background: *Non-destructive testing (NDT) is one of the most prominent techniques used in the conditional assessment of man-made structures. The data acquired from NDT is often vast and requires expert knowledge to detect faults. This can be costly, time consuming and often subjective. Artificial Neural Networks (ANNs) have been used to detect defects intelligently from large datasets gathered from NDT, but it would appear that their use is somewhat limited in detecting areas of corrosion in reinforced concrete.*

Aim: *This study aims to establish what evidence is available for the use of ANNs and/or alternative intelligent methods in the NDT of structures and more particularly*

reinforced concrete.

Method: *A mapping study of the literature was performed.*

Results: *179 papers were found and relevant data was extracted. It was found that feedforward ANNs, trained using the back-propagation algorithm, were the most popular artificial intelligent technique used in NDT applications. Simulations and experiments were found to be the most common study type, while finite element modelling which modelled the response of a structure under certain conditions was the most widely used data collection technique.*

Conclusion: *The results show a lack of studies which apply techniques to real-world applications that could be the main use of NDT techniques and the data processing which follows. It could be argued that the popularity of the simple feedforward ANNs could be due to simplicity and alternatives could be better in particular applications. A further investigation should assess the most suitable type of ANN, alternative artificial intelligent (AI) technique, or hybrid combinations of the two for NDT applications which will give the best detection results. The study has also confirmed a void in the use of ANNs to detect corrosion in reinforced concrete.*

B.1 Background

With an increasingly ageing building infrastructure, the detection of damage in built structures in the UK and around the world is becoming ever more important. In 2003, the UK concrete repair industry was estimated to be greater than 3% of the entire construction industry, at a cost in excess of one billion pounds [8]. One of the main areas in the damage detection field is that of non-destructive testing (NDT). NDT involves using non-invasive techniques to gather data on the condition of a man-made structure, which can then be used to make an assessment of its health. The data

gathered from NDT techniques can be vast and contain large amounts of noise, which has traditionally required expert analysis. An alternative approach when analysing data is to use Artificial Neural Networks (ANNs) which are particularly well suited to identifying patterns and perform well with noisy data. While ANNs have been used to detect structural damage looking at particular features of the structure (for example, a reduction in stiffness) and/or using numerical models, it appears that little evidence is available to support their applicability in detecting corrosion within reinforced concrete.

The methodology behind this mapping study has stemmed from the work conducted by Kitchenham et al [348, 69, 349] on evidence based procedures, more specifically, systematic literature reviews. The work carried out by Kitchenham et al [348, 69, 349, 350] has been adopted from the medical and social sciences context [70] and applied to the area of Software Engineering. It will be interesting to observe the extent to which the guidelines produced from this work will be applicable to the area of applied Artificial Intelligence in an Engineering context. A systematic mapping study is designed "to provide a wide overview of a research area, to establish if research evidence exists on a topic and provide an indication of the quantity of the evidence" [69]. A mapping study was performed as a precursor to a systematic literature review as a means of identifying the literature available for ANNs and their use in NDT, in order to identify the current literature, evaluate what has been achieved in terms of intervention and identify areas which can be improved upon which will guide the direction of future research. Any findings which arise as a result of this mapping study and a later systematic literature review will be reported where applicable.

As a mapping study is a preliminary, broader study than a comprehensive systematic literature review, this short report outlines the data collection process, presents a

summary of the results and discusses initial conclusions drawn from the data. The results found from performing a mapping study can be used as part of the foundation for a systematic literature review once areas of potential research have been identified. Section B.2 provides a brief introduction to the mapping study, outlines the study that was performed and presents the results. Section B.4 gives an analysis of the data and draws some initial conclusions.

B.2 The Mapping Study

Following the guidelines [69] and previous empirical studies [348, 349, 351], a protocol [352] for the mapping study was created which set out how the study would be performed and what data would be gathered as a result. One area of the protocol listed the research questions that the study would answer. These included:

1. Which bibliographic resources include papers on ANNs in structural assessment, more specifically NDT of reinforced concrete?

2. Which type of damage detection applications are ANNs or alternative intelligent methods mostly used in?

3. What is the most popular intervention technique?

4. Which study types are the most popular?

In the case of mapping studies and systematic literature reviews, the term intervention refers to the "software methodology/tool/technology/procedure that addresses a specific issue" [69]. For example, the intervention for one study may be the use of a probabilistic NN with thermal imaging equipment in order to assess the health of a structure.

B.3 Conduct of the Study

The study was conducted by the PhD student with two supervisors offering feedback on the planning, data collection and study review process. As outlined in the protocol [352], the scope of the study was as follows:

- **Population**: Man-made structures, particularly reinforced concrete structures.
- **Intervention**: Experiments and studies involving damage detection approaches using ANNs or alternative techniques and processes
- **Outcomes of relevance**: Quantity and type of evidence relating to various ANNs (or alternative(s)) used to detect damage techniques and processes
- **Experimental Design**: Any scientific experiment or study of the searched papers.

A number of electronic resources were used in order to gather the required data which included Google, ASCE, IEEE Digital Library, ACM Digital Library, ScienceDirect, Blackwell-Synergy and NDT.net. Again, as a mapping study is a broad search for literature, no publication date limit was applied to the search in order to collect and evaluate over an indefinite time period.

The search term outlined in the protocol was first piloted on Google to evaluate its performance in acquiring relevant papers. When used on Google it appeared that the search term worked well and it was decided at this stage that the original search string would be suitable for the search process. The search term used was:

> neural networks OR artificial neural networks OR ANN OR intelligent data processing system AND rust detection in reinforced concrete OR struc-

tural health monitoring OR non-destructive testing OR NDT OR damage detection

At first glance, the search string appeared to be successful as it returned 66 relevant papers on Google and 29 papers using ScienceDirect. After this initial success, however, the search string was found to be less successful in returning relevant papers using other electronic resources. This is due to the fact that some online resources require specific syntax in order for a search to be successful and some also have a limit on the number of characters that can be entered as a search string as reported by Brereton et al [348]. One could argue that the search term should have been piloted first on all resources before being used to search for papers on the resources for which it was successful. However, this problem with search terms was not foreseen at the beginning of the study due to the lack of experience from the research student. It was decided to keep the papers collected from using the original search string and to use combinations of it for resources which returned poor results. This may have led to some unfair results as some papers were found with slightly different search strings than others, although it has been found to be quite common to have to adapt search strings to different search engines, as many do not support systematic literature reviews [69]. The other reason for keeping the results was the amount of time and effort which had gone into finding and documenting the publications already found via Google and ScienceDirect. Table B.1 shows a summary of the most successful adapted search terms used for each of the remaining resources.

Table B.1: A summary of the most successful search terms for each of the resources.

Search string	**Resource**	No. papers returned	Relevant papers
"neural networks" AND (structural health monitoring OR rust detection reinforced concrete OR non destructive testing OR damage detection)	ACM	200	3
('neural networks' <and> (structural health monitoring <or> rust detection reinforced concrete <or> non destructive testing <or> damage detection) <in> (pdf-data,metadata))	IEEE	277	20
"neural networks" AND "structural health monitoring"	Blackwell-Synergy	10	5
"neural networks" AND "structural health monitoring"	ASCE	6	5
"neural networks" AND "non destructive testing"	NDT.net	94	27

As Table B.1 shows, the most successful adapted search string for each resource had varied success. For example, the most successful search term used on NDT.net returned 27 relevant papers, while the most successful search term of ACM returned only 3. The study used the following inclusion criteria:

- books, papers, technical reports, web articles and grey literature that outline ANN or an equivalent alternative approach applications in structural assessment
- where several independent studies are reported in the same paper, each relevant study will be treated as an independent primary study

and the following exclusion criteria:

- studies that do not use intelligent data processing techniques in NDT structural health applications
- studies that do not apply NDT techniques in the assessment of structural health
- literature that is only available in Slideshow format

It was decided for this initial study that articles which were only available in abstract form would be included because an abstract should provide enough information regarding the study to be sufficient. It was later found that this was not the case as many abstracts were poorly written as has been reported in [353]. In these instances, the author was contacted to gather the required information for data collection.

B.4 Results

From the search strings outlined earlier, a total of 191 papers were found, which were reduced to 179 after some papers were found to be irrelevant, lacking sufficient detail

and, in one case, a duplicate of a paper already found. Data was extracted according to the protocol [352] and stored in a spreadsheet. The protocol [352] outlined the following data to be extracted and analysed:

- publication type vs. number of studies
- resource type vs. number of studies
- neural network/alternative intervention vs. number of studies
- type of training algorithm used if ANN was used
- NDT technique vs. number of studies
- damage detection applications vs. number of studies
- experiment/study type vs. number of studies

Originally, intervention was specified in the protocol [352] as one broad bullet point. It soon became clear, however, that it would be unfeasible to classify a paper's intervention based on both the type of NDT and artificial intelligent (AI) technique used. This would have led to multiple classifications of intervention that would be difficult to analyse. For example, one paper may have used an ultrasonic NDT combined with a standard feedforward ANN trained using the backpropagation algorithm. Another paper could have used the same NDT technique, but with a probabilistic ANN, which would have resulted in two completely different intervention classifications. For this reason, it was decided to split intervention into two types of intervention, AI intervention and NDT technique intervention. It was also decided later on in the study to split the AI intervention to account for the type of training algorithm used. Table B.3 shows the initial publication type of the studies collected, whilst Table B.4 shows the

classification of the top five resources from which the papers were obtained. For the full table, see the full report [68].

Table B.3: Papers classified by publication type.

Publication type	**No. of studies**
Journal	111
Conference	74
Research project/workshop	5
Poster	4
Magazine	2
Web article	1
Masters thesis	1
Unclassified	4

Table B.4: Papers classified by resource acquired from.

Resource	**No. of studies**
Google	66
NDT.net	49
Sciencedirect	34
IEEE	20
n/a	18
Blackwell-Synergy	17

Table B.3 shows that the most common types of papers found were in journals and conference papers which is not surprising since most research is published in these two media. Table B.4 shows that, of the subject-specific resources, NDT.net was the best resource to search for materials on NDT techniques and intelligent data processing as it returned the most relevant materials. Again this is not surprising since the website specialises in NDT and its applications (note, Google is not a subject specific resource as it performed a broad scan on all resources). Google returned the highest number of papers, which is expected due to the broad search it performs across all resources while searching on subject specific resources only returns results from the resource's database. These two comparisons are very basic in the sense that they show the most common type of publication and the resource which is most likely to return a relevant paper which are superficial results. The more interesting results arise from the analysis of the intervention used and the application to which it was applied. Tables B.5, B.6 and B.7 show the intervention used in terms of the top ten ANN type or alternative AI used, the top five training algorithms used if a multi-layer ANN was used, and the top ten NDT techniques used, respectively. For the respective full tables see the full report [68].

Table B.5: The ten most popular types of ANN/alternative used in NDT domains.

ANN/alternative used	**No. of studies**
Feedforward MLP	95
Hybrid system	11
Probabilistic ANN	9
Radial Basis Function ANN	8
Wavelet ANN	7
SOM	7
Fuzzy ANN	5
Fuzzy logic	5
Counter propagation ANN	4
Emulator ANN	4

Table B.6: The five most popular ANN training algorithms used.

Training algorithm	**No. of studies**
Backpropagation	69
Levenberg-Marquardt	18
Resilience propagation (RPROP)	3
Adaptive random search algorithm	2
Parzen window method	1

Table B.7: The ten most popular NDT interventions.

NDT technique	**No. of studies**
Finite Element Model (FEM)	37
Vibrational response	35
Ultrasonic testing	24
Eddy Current testing	16
Impact response	11
Reduction in elemental stiffness	10
X-ray	10
Photographic/video images	9
Frequency response functions	7
Acoustic emission	6

Table B.5 shows that when ANNs were used in an NDT application, the vast majority of studies used the simple feedforward multilayer perceptron neural network. The table also shows that fuzzy logic was the only alternative technique to get into the top ten, which could suggest the popularity and suitability of ANNs in NDT applications. When ANNs were used in NDT applications, the back-propagation algorithm was used significantly more often than other training algorithms as Table B.6 shows. Table B.7 shows that the NDT techniques extracted from the papers were much more varied than the previous two classifications of intervention. Whilst finite-element modelling (FEM) was found to be the most widely used, methods such as vibrational response and ultrasonic testing have also been used in quite a number of studies. Table B.8 shows the ten most popular application domains of the aforementioned interventions.

Table B.8: Top ten NDT applications.

Application	**No. of studies**
Bridges	25
Beams	15
Structures/buildings	13
Frames	12
Composite structures	12
Weld defects	11
Pipelines	9
Aeronautical applications	9
Aluminium panels/plates	7
Trusses	7

As Table B.8 shows, the number of NDT applications is widespread. From the papers found, there were a total of 57 different applications, many of which were found in only one publication. For the full table, see the full report [68]. The most commonly used application was bridges, with the remaining nine most popular applications within close proximity of one another. The final classification of papers was the study type of the papers found, which is shown in Table B.9. Papers were classified into five categories. Each category is defined below:

- simulation - study involves using a numerical model based approach to model the reaction of a structure to some other force or change in parameter(s),
- experiment - study uses a test piece in a controlled lab environment to gather

data using the proposed NDT technique,

- case study - study uses a real world structure as the focus of data collection,
- literature review - presents an outline of publications produced in the field of NDT and ANN or alternative AI technique,
- proposal - study presents an outline for a new technique that is yet to be developed.

Table B.9: Papers classified by study type.

Application	**No. of studies**
Simulation	79
Experiment	78
Case study	31
Literature review	5
Proposal	5
Unknown	10

B.5 Analysis and conclusions

Before considering analysis of the results, threats to validity of the mapping study results must be considered. One threat is ensuring that all the relevant publications have been identified. It could be possible that relevant papers may have been left out of the study. This situation may arise because of the use of a poor search string. However, the search string was trialled on two of the main resources in order to achieve the best

coverage and avoid this situation. The most likely reason that some papers may have been missed is the way in which certain search engines handle search strings, which has been highlighted previously [69, 348]. One could argue that the reason for the very few alternative AI methods found compared to ANN is because the search string contained the terms "ANN" and its derived terms, but not specific alternative terms such as "fuzzy logic". This is one problem with the search string. It is, however, difficult to overcome as the number of alternative techniques is extensive. If every alternative were to be included in the search term it would become very large, which would lead to the degradation of the search string's general performance. A possible solution to this problem could be to include the most popular alternatives in the search string, some of which have been identified as a result of this study. This will be investigated further in the systematic literature review to be conducted at a later date.

The other threat to validity could be that the classification used in data extraction may be biased and unreliable. This could occur as a result of the classification process taking place over a number of weeks, where one technique used in one paper may be classified differently to a paper using the same technique which was classified on a separate day. This problem is even more likely with this study as the extraction and classification was conducted by the sole PhD student. This problem was overcome by giving the supervisory team several papers selected at random to extract and classify data. The data extracted was then compared to the data collected by the student, with both versions matching satisfactorily. This check also helped to counteract bias as classification can be subjective due to opinions regarding classification differing. In order to check that all the studies which were selected for data extraction were relevant to the study, the test-retest [69] approach was used. This involved selecting ten random papers to ensure that they met the inclusion and exclusion criteria before

being analysed. All papers selected at random passed this test.

From this initial study, the following points can be made from the data,

- The resources which returned the most relevant publications were found to be Google and NDT.net. As mentioned earlier this is probably due to the fact that Google performs a widespread search and NDT.net specialises in NDT techniques and their applications.

- The simple feedforward multilayer perceptron (MLP) ANN was by far the most commonly used data processing technique. As mentioned earlier, fuzzy logic was the only alternative AI approach outside of ANNs to be listed in the ten most popular techniques. This could be because ANNs are the most prominent data processing technique in NDT applications. It may also be due to the limitation of the search string as mentioned earlier. For the later systematic literature review, some of the most common alternatives found from this study may be included in the search string to counter this problem.

- By some distance, the back-propagation algorithm was the most widely used technique for training MLP ANN. It could be argued that in many studies, ANNs were used as a black box without the authors having any background knowledge into the workings of ANNs and related techniques. The fact that feedforward MLP ANN and back-propagation were used so extensively could be due to their simplicity and ease of use, as Adeli [140] points out. Ease of use, however, may not be the best characteristic to evaluate when choosing a type of ANN or alternative method to perform data analysis. For instance, the back-propagation algorithm has many shortcomings, including slow learning rate and a convergence rate which is highly dependent on the learning and momentum parameters chosen [140]. If

these parameters are chosen poorly, the ANN can get stuck in local minima which negatively impact on the convergence rate and the performance of the ANN. For these reasons some of the studies' results could have been improved had they used a more suitable ANN or alternative.

- Experiments and simulations were the two study types which were most prominent from the relevant publications. This finding is closely linked to the next point, as FEMs are types of simulations and the majority of the vibrational techniques were observed during experiments. The number of case studies shows a lack of projects which apply their techniques to real world applications, which is ultimately towards what they should be geared.

- Of the NDT techniques used, FEMs were found to be the used the most. This usually involved creating a numerical model which was changed in some form (for example, a reduction of certain parameters). The models' reaction to these changes was observed and recorded. The problem with using FEMs, however, is that they involve numerical equations to simulate some properties of a structure. While studies show promising results using FEMs, the techniques they propose could perform much worse in the real world due to factors like uncertainty and noise. Vibrational response techniques were also found to be widely used.

Overall, it would appear that there is a definite lack of studies which apply proposed techniques to real-world applications. Applying new techniques to real world applications should be a priority for studies as the true success of the technique can only be assessed by its reliability and accuracy in the real-world. The mapping study has also shown that the simple feedforward multilayer perceptron are the most widely used type of ANN, which is probably due to their simplicity and ease of use. A further investi-

gation will look into finding the most suitable type of ANN, alternative AI technique, or hybrid combinations of the two for NDT applications. This study will also search for alternative techniques more fairly by improving the representation of alternative techniques within the search string. One of the most prominent findings of this study is that there appears to be a total lack of publications with the aim of detecting corrosion within reinforced concrete structures. Not a single publication found from the mapping study discussed detecting rust or corrosion within reinforced concrete. One reason for this could be that the authors chose not to disclose the type of defects which they sought in order to protect their research. However, one could argue that despite this possibility, other defects such as cracks and delaminations were discussed and if rust had been detected, it would have been mentioned at least once in the many papers found. This apparent lack of corrosion detection using NDT techniques and ANNs is a major problem which needs to be addressed by future research. As a final remark, there appears to be very little literature or guidelines on performing mapping studies compared to that on systematic literature reviews. In order to improve the process and reliability of such studies, guidelines would be very beneficial.

B.6 Acknowledgements

We would like to thank Dr. Mark Turner for his help and advice regarding the mapping study process.

Appendix C

Determining the spatial extent to which defects could be detected

This section details the analysis conducted in order to determine the spatial extent a break in an intersection of rebars (defect 5B) could be detected by the EMAD. In Chapter 5, data collected from scan lines four, five and six were presented where it was shown that the signature of 5B could be detected in these scan lines. Figures C.1, C.2, C.3 and C.4 show the data collected from scanning rebars three and seven before and after the break had been created.

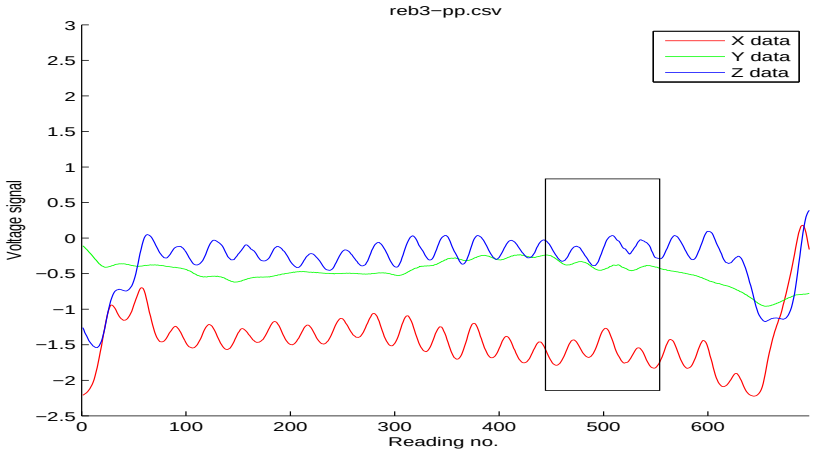


Figure C.1: Data collected before break insertion along rebar three. The black box indicates the expected location of defect 5B should it be visible from scan line five (see Figure C.2 below).

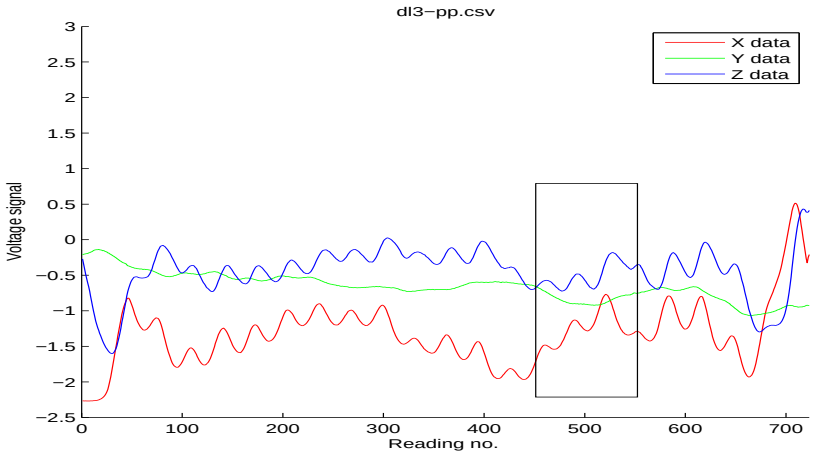


Figure C.2: Data collected after the break along rebar three with the signature of defect 5B shown by the black box.

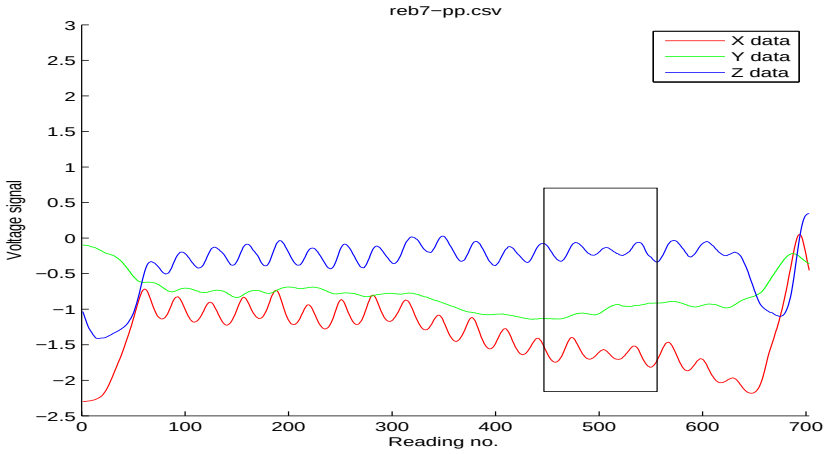


Figure C.3: Data collected before break insertion along rebar seven. The black box indicates the expected location of defect 5B should it be visible from scan line five (see Figure C.4 below for the post-break insertion data).

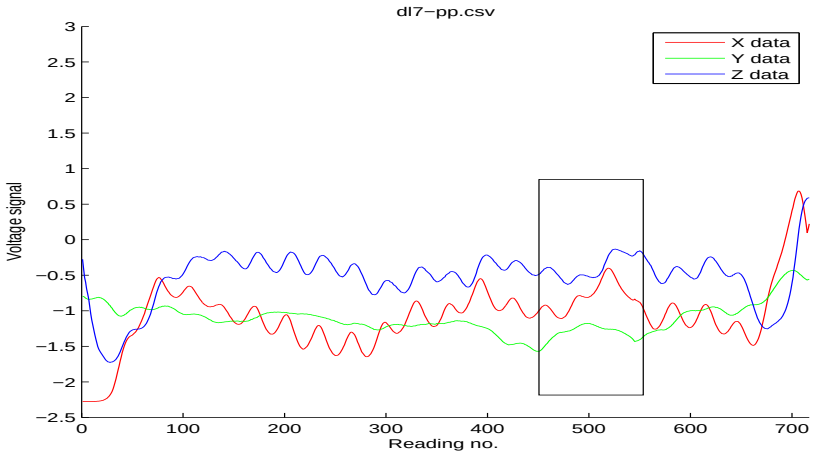


Figure C.4: Data collected after the break along rebar seven with the signature of defect 5B visible as highlighted by the black box.

When comparing the pre and post-break plots, it is clear that the probe still detects the break a little, as the data changes around the 500^{th} datapoint as before. Therefore, scan lines two and eight were analysed to evaluate the magnetic strength of the break further away. Figures C.5, C.6, C.7 and C.8 show the pre and post-break data for scan lines two and eight.

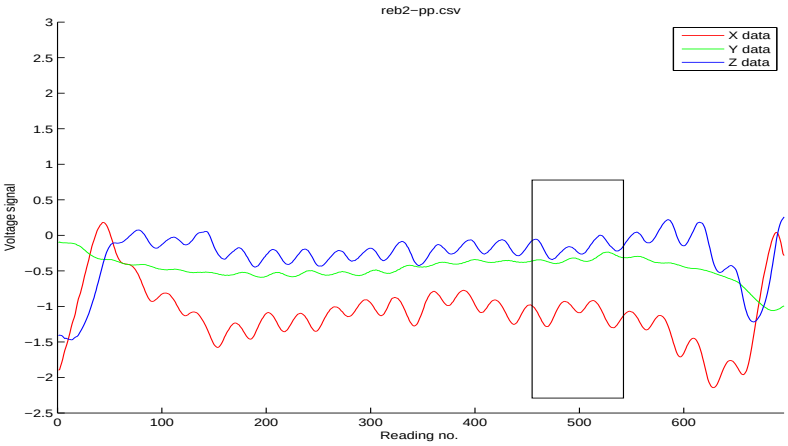


Figure C.5: Data collected before the break along rebar two with the black box indicating the location of the break signature should it be visible after insertion.

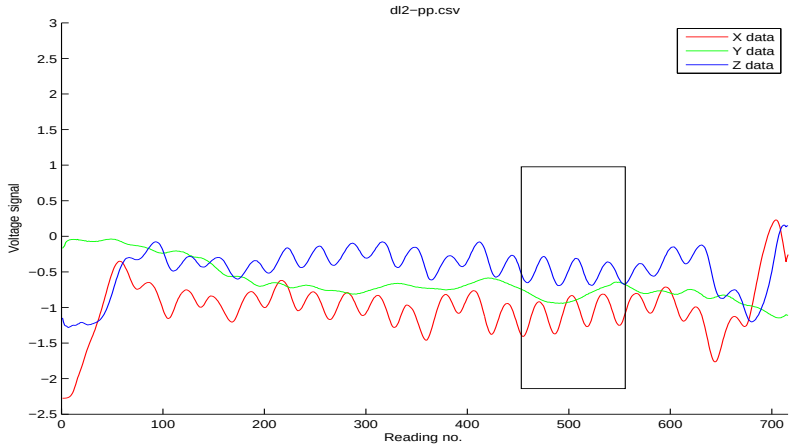


Figure C.6: Data collected after the break along rebar two with the location of defect 5B shown by the black box. Comparing this figure with Figure C.5 reveals little change in the data after the break insertion.

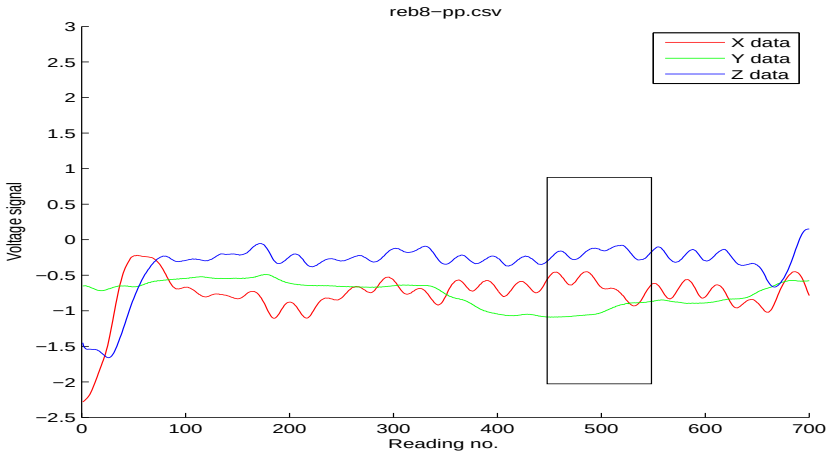


Figure C.7: Data collected before the break along rebar eight with the expected location of the break defect highlighted.

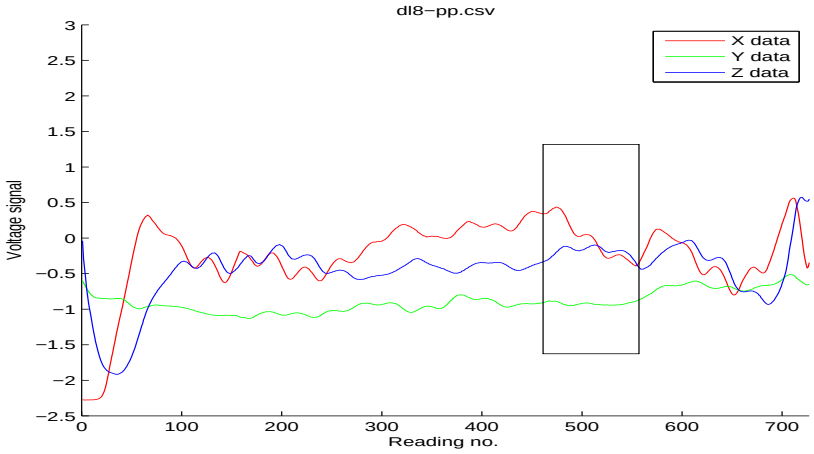


Figure C.8: Data collected after the break along rebar eight with the location of the break defect highlighted.

As Figures C.5, C.6, C.7 and C.8 show, the effects of the break almost fade away along rebars two and eight, with a slight change in signal still present along scan line eight. Therefore, the inserted break could still be detected up to three rebars away transversely, albeit a small signal. As the mesh was due to be set in concrete, the distance between the probe and the steel would be increased as a result of an added layer of concrete as well as the width of the boards placed over the concrete during scanning, both of which would reduce the signal strength of the defect. The break inserted into the mesh was also at an intersection of rebars which creates the largest signal as a result of larger fringing fields caused by four broken rebars. Other defects such as corrosion on breaks on straight sections of rebar would produce smaller fringing fields. With this in mind, an exclusion zone 1.5 rebars square was placed around each defect.

Appendix D

Introducing corrosion products onto the reinforcing mesh

When steel is exposed to the atmosphere, often corrosion products are created as a result of oxidation, generally causing the colour to change from a shiny metallic appearance to a rusty orange. In this case the main corrosion product present is lepidocrocite. In order to verify that this was the main iron oxide present on the mesh prior to magnetite and maghemite creation, iron filings scraped from some rebars of a similar mesh were analysed using a Bruker D8 Advance x-ray diffractometer (XRD) with a CuK_α source and a scattering angle (2θ) between 5° and 120°. The filings were ground into a fine powder and placed in a sample holder inside the XRD. During the data collection the sample holder was rotated continuously to average over all crystallite orientations.

Figure D.1 shows the results obtained using XRD on oxidised rebar filings. The black plot is the XRD signature of the oxidised iron filings, whilst the coloured vertical lines indicate the calculated intensity of the diffraction peaks from the different iron

oxides (taken from Cornell and Schwertmann [344]). The peaks occur as a result of interference caused by x-rays scattered from atomic planes within a crystallite. Where the peaks of the iron filing XRD signature match the intensity peaks of the different iron oxides this indicates that that particular iron oxide is present within the sample.

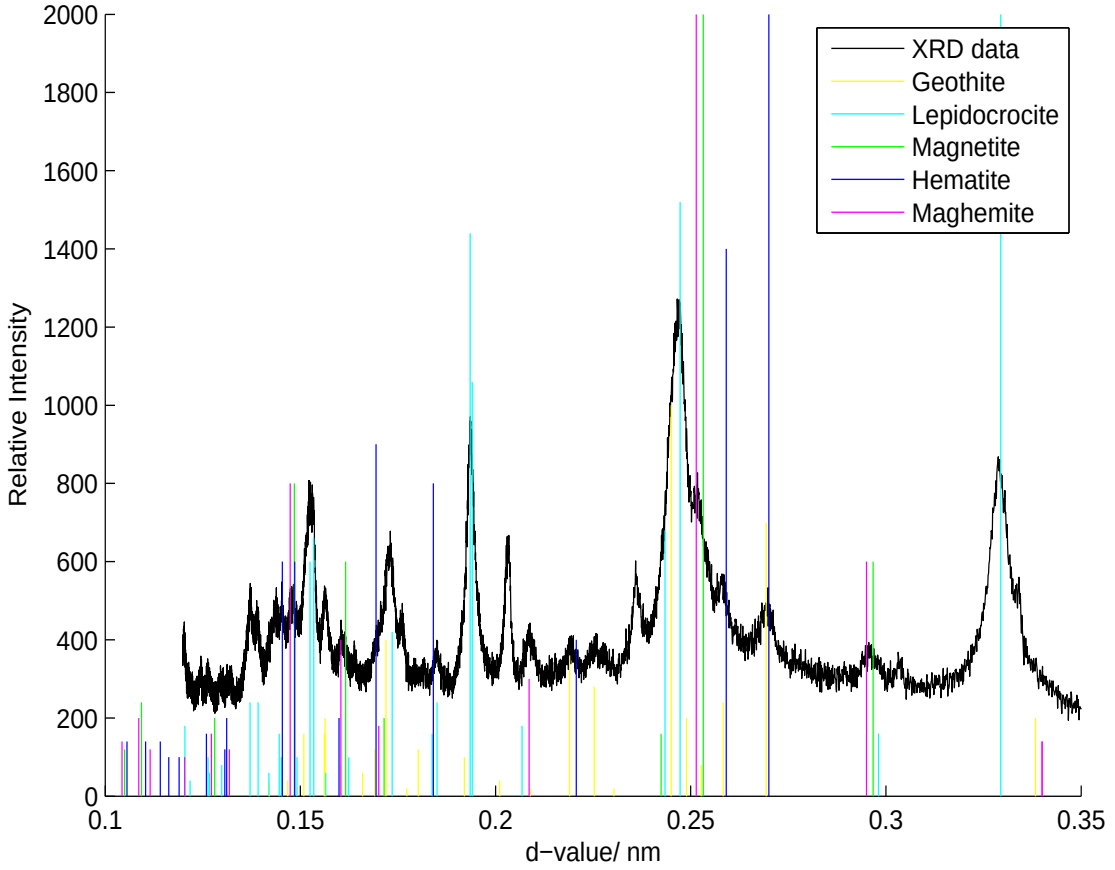


Figure D.1: XRD spectrum from the oxidised iron filings collected (black line). The lines relate to the intensity peaks of the different iron oxides.

Inspection of the X axis of Figure D.1 shows the spectrum plotted as a function of a d-value/nm. The value of d was calculated according to:

$$d = \frac{n\lambda}{2\sin\theta} \tag{D.1}$$

where d is the spacing between a set of crystallite planes, λ is the wavelength of CuK_α radiation (which is 1.5418 Å) and n is the diffraction order which is assumed to equal one.

As the graph in Figure D.1 shows, the majority of the peaks of the sample correspond closely to the intensity peaks of lepidocrocite (cyan peaks), meaning that the main iron oxide present is lepidocrocite. Smaller peaks indicate small amounts of maghemite (for example, see magenta peaks around d-values of 0.25 nm and 0.3 nm), which indicate small amounts of these products are also present. The peak at around 0.2 nm, which does not match any of the iron oxide intensity peaks, could be a contaminant product such as some unoxidised iron which may have fallen into the sample during collection. In summary, the main iron oxide product present in the mesh after exposure to the atmosphere was lepidocrocite, with small amounts of other iron oxides such as maghemite.

In order to obtain the desired magnetic iron oxide products of maghemite and magnetite, the target areas of the mesh where corrosion was to be created were heated using a heat gun in the range 250-290 °C for a period of 10 minutes. This procedure was adopted based on the work conducted by Gehring and Hofmeister [331] who found that heating lepidocrocite in the range 200-300 °C created maghemite; that was then confirmed by infra-red spectroscopy. The authors found that maghemite was formed after heating lepiodocrocite in air for one hour at 250 °C. It was also found that a

temperature of 273 °C created maghemite with the strongest magnetic field, whilst temperatures over 300 °C led to a decreased magnetisation in the material. This is shown in Figure D.2.

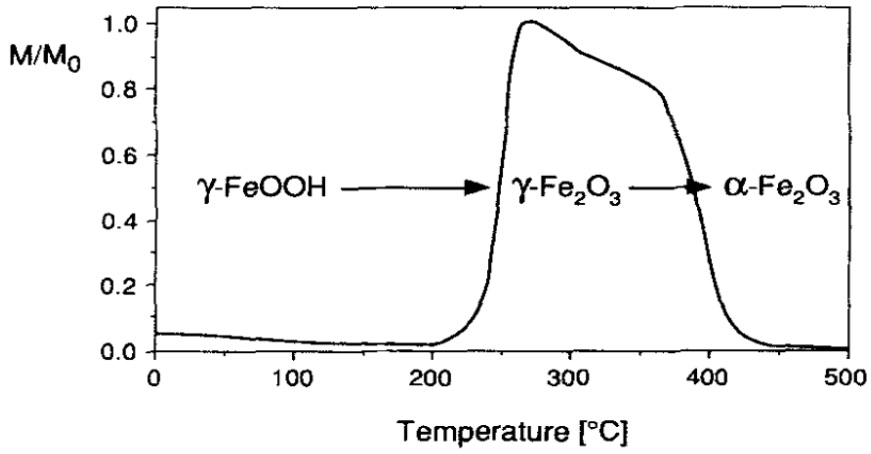


Figure D.2: A thermomagnetic curve of lepidocrocite in an external magnetic field of 0.2 mT when heated from 0°C to 500° (reproduced from Gehring and Hofmeister [331]).

As Gehring and Hofmeister heated small samples in a controlled laboratory oven, it was possible to heat them at a constant temperature for one hour. This approach was not feasible with the reinforcing mesh owing to its size. Hence the target areas were heated to the set temperature range for 10 minutes so that the maximum magnetisation creation was possible. In order to reduce the variability of the temperature of the steel, the heated area was wrapped in several layers of aluminium foil with a temperature sensor inserted inside to monitor temperature changes. Initial experiments were conducted using small rebars which were heated to the set temperature range for the set time period. Once this was complete, the corrosion products formed on the steel were extracted and analysed again using XRD. Figure D.3 shows the results.

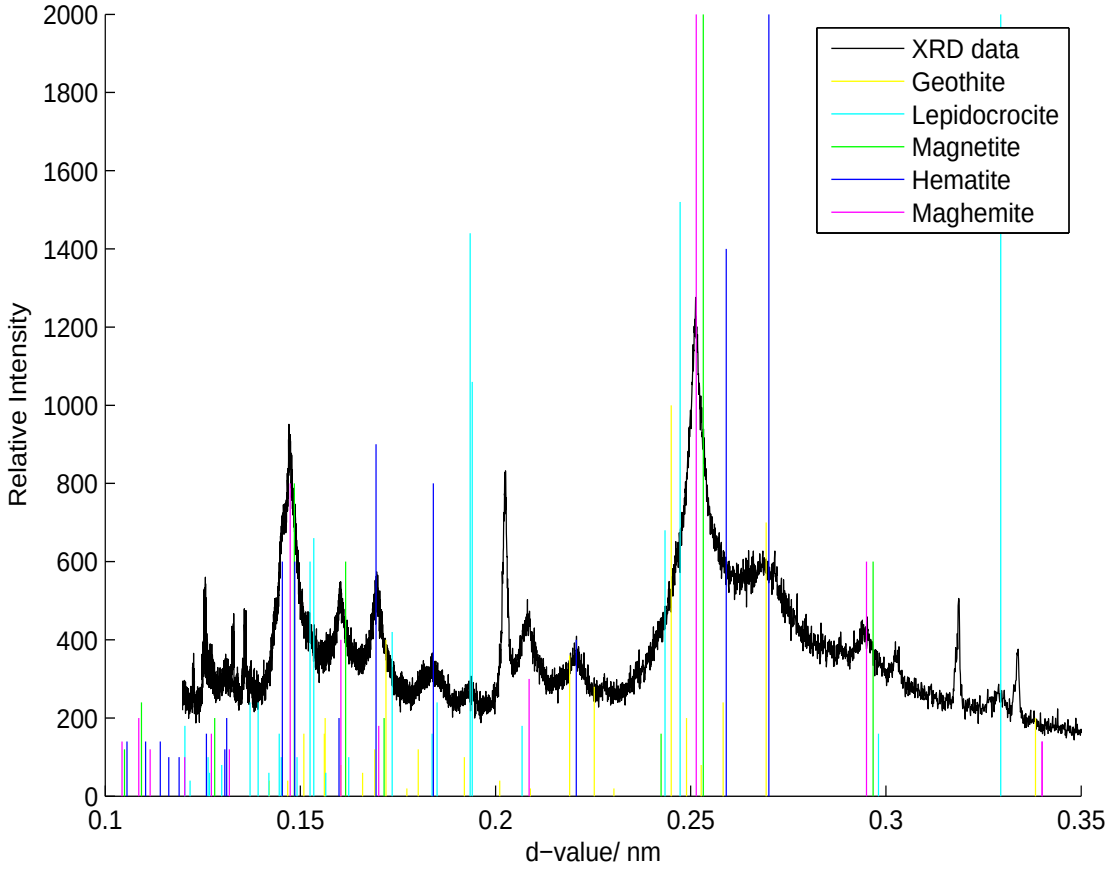


Figure D.3: XRD results from oxidised iron filings collected (black line) after heating between 250-290°C for 10 minutes. The lines relate to the intensity peaks of the different iron oxides.

As Figure D.3 shows, the major iron oxides had now changed from lepidocrocite (cyan lines) to maghemite (magenta lines). This is most apparent around 0.25 nm, where the biggest peak of the XRD line moved slightly to the right after heating, aligning with the maghemite line position. This peak is also very close to the magnetite intensity peak (green) so it is likely that there is magnetite present also. Almost all of the lepidocrocite has disappeared, with only a few intensity peaks matching the lepidocrocite lines. One can also see that there is some hematite present in the sample (see the XRD plot and blue lines between 0.15 and 0.2 nm). The unclassified peak is still present in the post heated iron filings plot around 0.2 nm, which reinforces the idea that this could be unoxidised iron particles. Figure D.4 shows the XRD spectra from both unheated and heated samples overlayed for comparison.

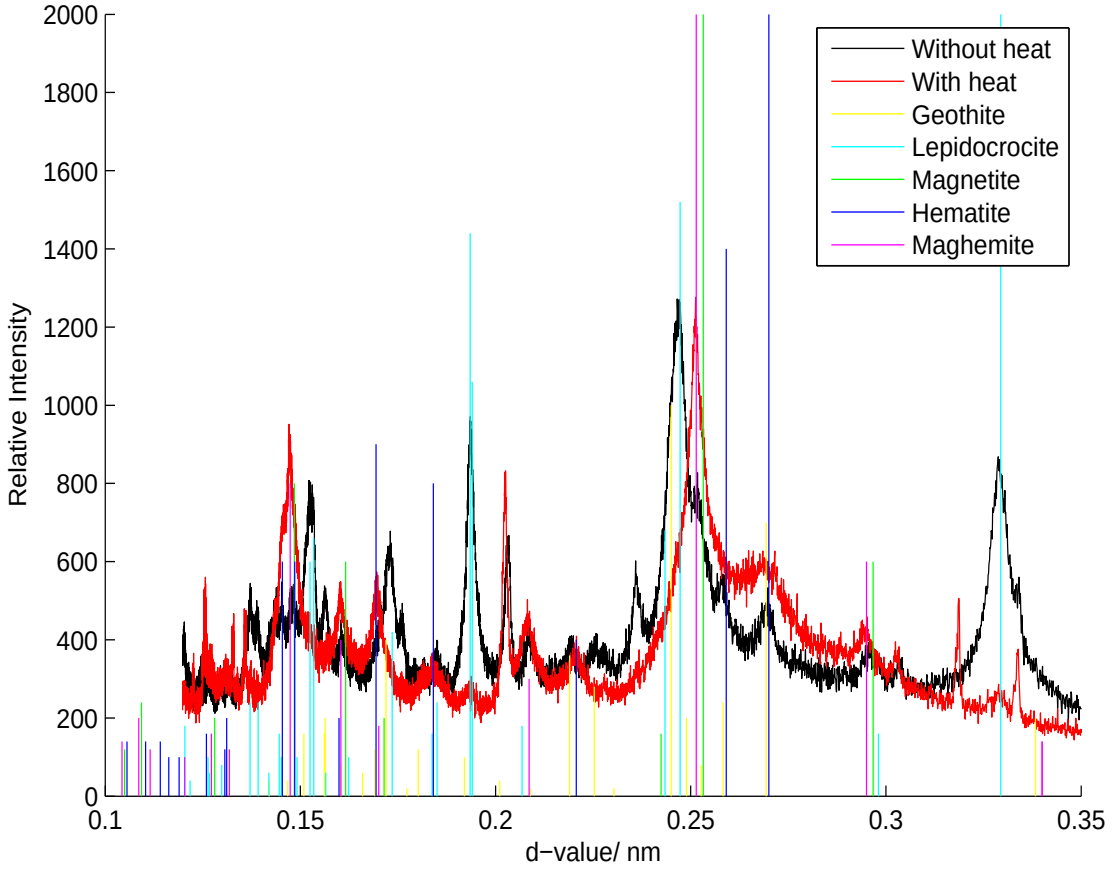


Figure D.4: A comparison of the non-heated iron filings (black line) and the heated iron filings (red line). Note the change in alignment between the two lines with the different iron oxide line intensities.

From these results, it can be said that the heating process successfully converted lepidocrocite to maghemite and magnetite. As this was a success, the areas of the mesh selected for corrosion creation were then heated using the same procedure, as was outlined in Figure 5.16 of Chapter 5.

Appendix E

Corrosion tank set-up

A corrosion tank was created to form a mould for the concrete and to provide a water-tight environment for accelerated corrosion. As the steel mesh was rather large, one of the only weatherproof locations in which it would fit was a greenhouse. The tank was then waterproofed with standard PVC pond liner, to ensure that no concrete escaped during the setting and curing process and that no sodium chloride solution escaped at later corrosion acceleration stages. Wooden ply boards were then placed over the concrete, after curing, which were marked with the appropriate paths for scanning.

The concrete was mixed using a cement mixer, consisting of Portland cement, building sharp sand and 10 mm aggregate, using the standard ratios of 1:2:4 by weight respectively. Guidelines from the ASTM C 192 standard [354] for making and curing concrete test specimens in the laboratory were followed for the mixing process. A water-to-cement ratio of 0.57:1 was used, following Wheat [332], creating concrete that was more porous than usual concrete mixes (typical water:cement ratios are roughly 0.35:1 to 0.4:1). This would enable corrosive agents to penetrate the concrete more easily, speeding up the corrosion process. To minimise variation in the consistency

of the concrete, all of the mixing was carried out during the same day, at the end of which the concrete was covered in water and left to cure for four weeks. This was to minimise the shrinkage of the concrete after setting, reducing the dangers of cracking and spalling which can occur as a result of the concrete contracting too quickly.

After curing, a procedure similar to the ASTM G109 standard [333] was adopted for accelerating ageing, in which the concrete was soaked in a salt water solution for a period of two weeks, drained and left to dry for two weeks. This cycle was then repeated for 57 weeks, with data collected using the EMAD once a week. The water-to-salt ratio was 97 parts water to 3 parts pure freeze dried sodium chloride by weight as specified by the ASTM G109 standard [333]. Maximum and minimum temperatures were also recorded using a max/min digital thermometer on the day of data collection in order to give an overview of the temperatures the concrete experienced during the experiment. As the experiment was located in a greenhouse, the temperature fluctuated greatly throughout the year, especially during extreme weather conditions. This made the temperature difficult to regulate which meant that the ASTM G109 standard's requirement of keeping the temperature in the range of 23 +/- 3°C was not met.

Appendix F

Scan lines of all defects pre-concrete encasement

This section details all scans collected when energising longitudinally and scanning longitudinally and transversely after all 12 defects had been introduced into the mesh. Defects which could be detected from neighbouring scan lines are also labelled.

F.1 Longitudinal energisation and scanning

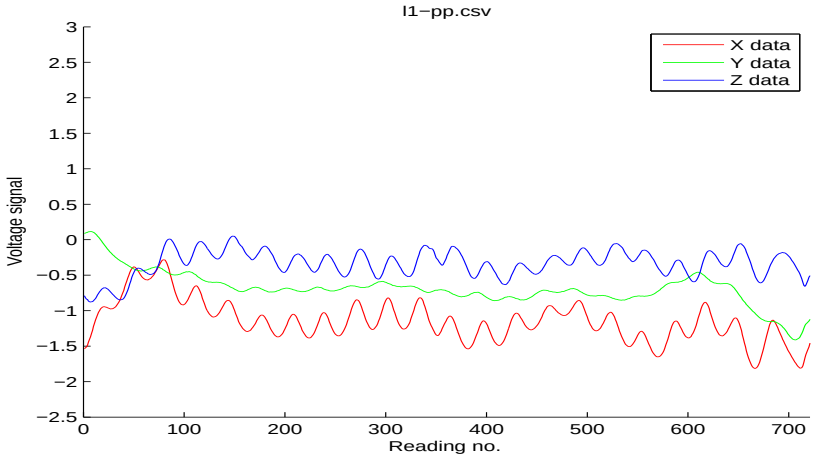


Figure F.1: Data collected when scanning longitudinally along scan line one after longitudinal energisation.

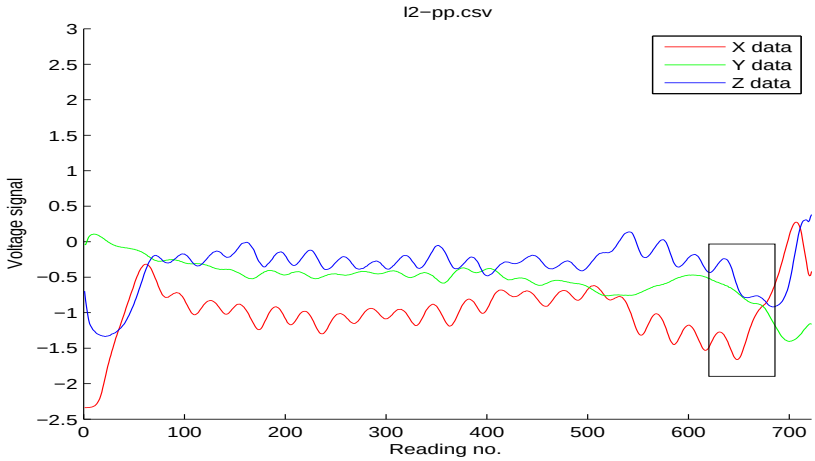


Figure F.2: Data collected when scanning longitudinally along scan line two after longitudinal energisation showing the location of defect 1C.

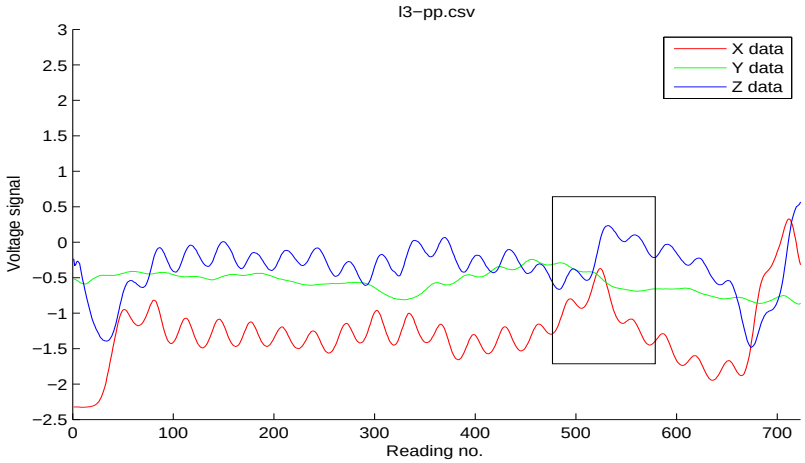


Figure F.3: Data collected when scanning longitudinally along scan line three after longitudinal energisation showing the signature of defect 5B present from scan line five.

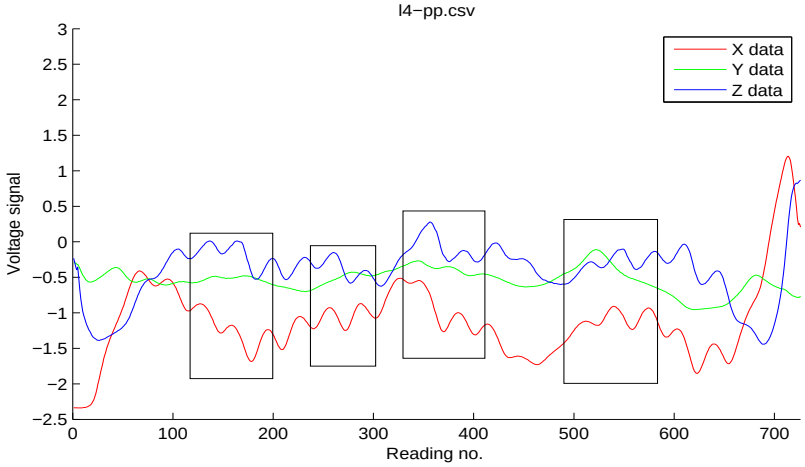


Figure F.4: Data collected when scanning longitudinally along scan line four after longitudinal energisation with defects 2C and 3C labelled, as well as defects 4B and 5B from neighbouring scan line five.

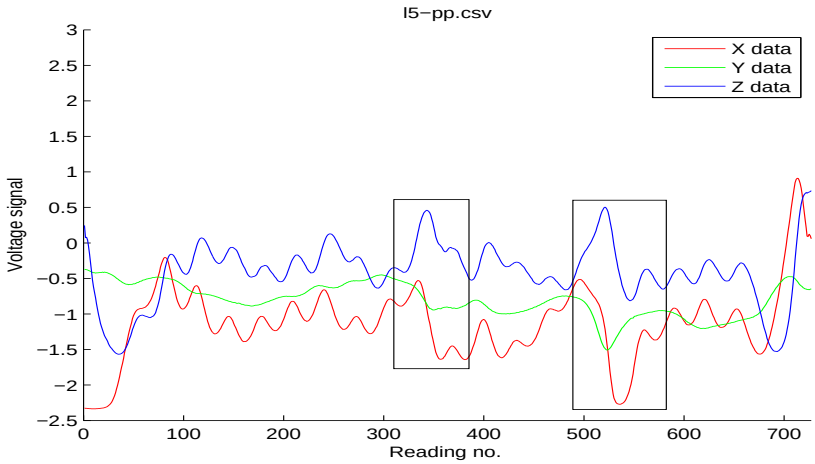


Figure F.5: Data collected when scanning longitudinally along scan line five after longitudinal energisation with defects 4B and 5B labelled.

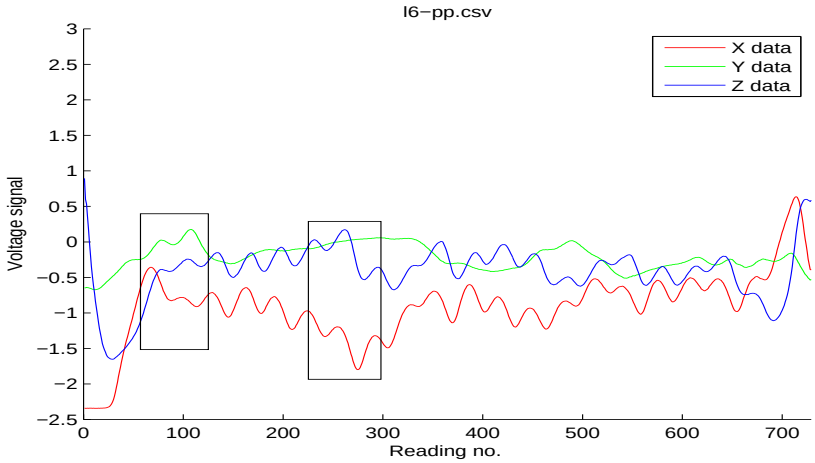


Figure F.6: Data collected when scanning longitudinally along scan line six after longitudinal energisation with defects 7BC and 8BC from neighbouring scan line seven labelled.

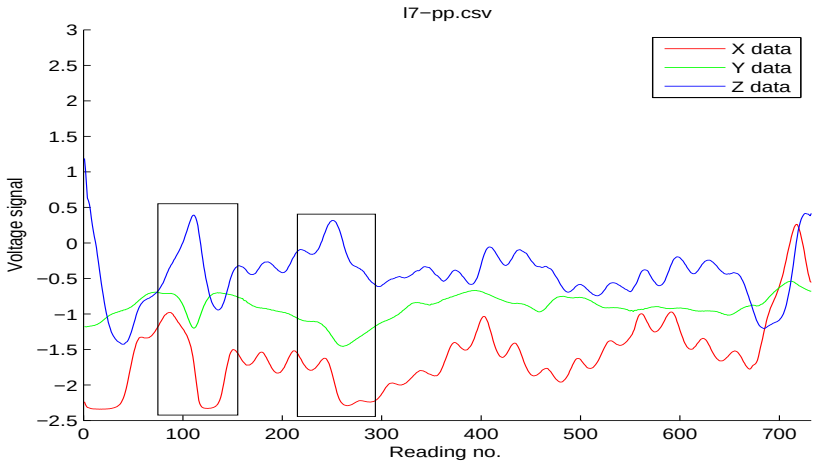


Figure F.7: Data collected when scanning longitudinally along scan line eight after longitudinal energisation with defects 7BC and 8BC labelled.

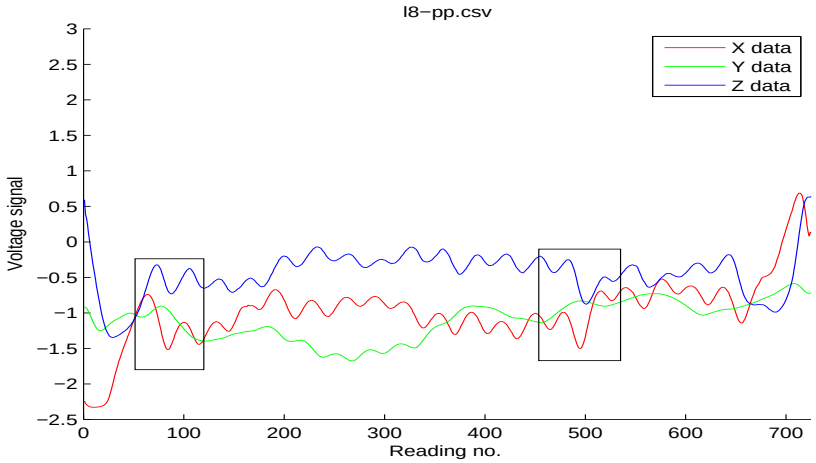


Figure F.8: Data collected when scanning longitudinally along scan line eight after longitudinal energisation with defects 7BC and 11BC visible from scan lines seven and ten labelled.

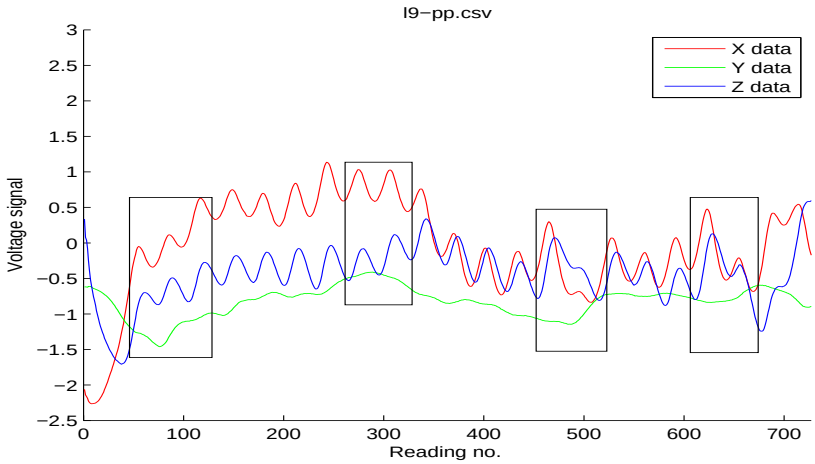


Figure F.9: Data collected when scanning longitudinally along scan line nine after longitudinal energisation with defects 9B, 10C, 11BC and 12B visible from scan line ten labelled.

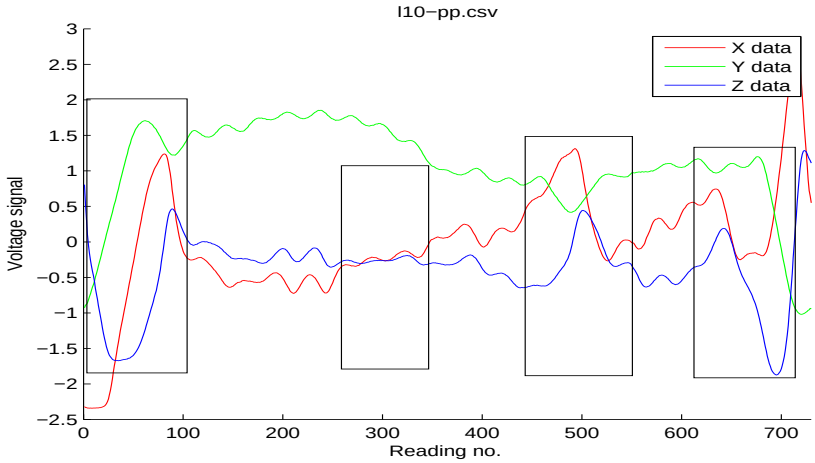


Figure F.10: Data collected when scanning longitudinally along scan line ten after longitudinal energisation with defects 9B, 10C, 11BC and 12B labelled.

F.2 Longitudinal energisation and transverse scanning

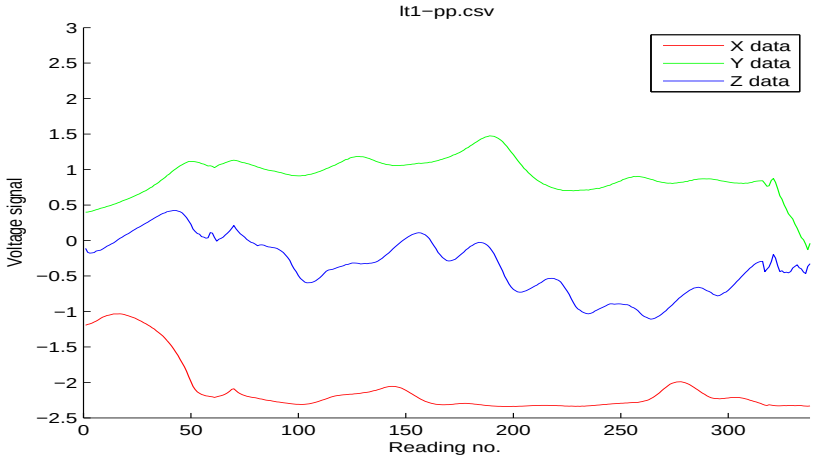


Figure F.11: Data collected when scanning transversely along scan line one after longitudinal energisation.

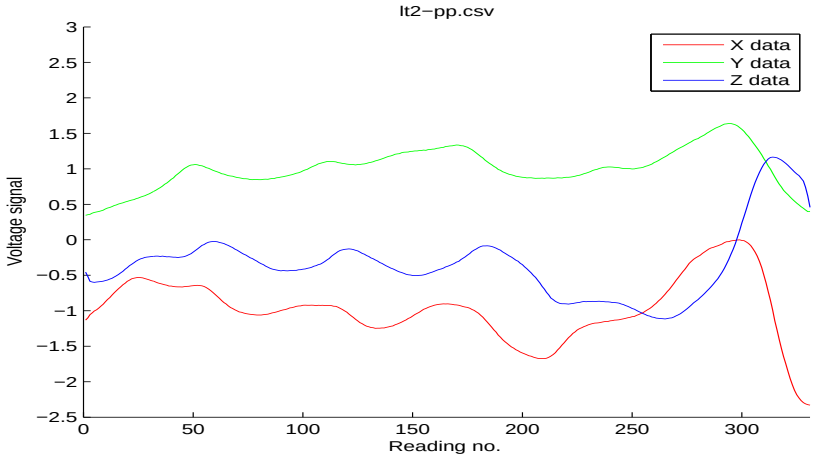


Figure F.12: Data collected when scanning transversely along scan line two after longitudinal energisation.

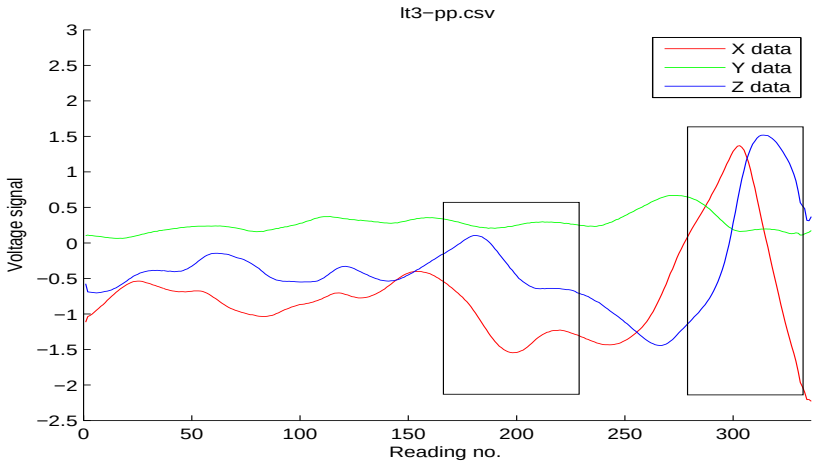


Figure F.13: Data collected when scanning transversely along scan line three after longitudinal energisation with defect 9B and defect 7BC from neighbouring scan line four labelled.

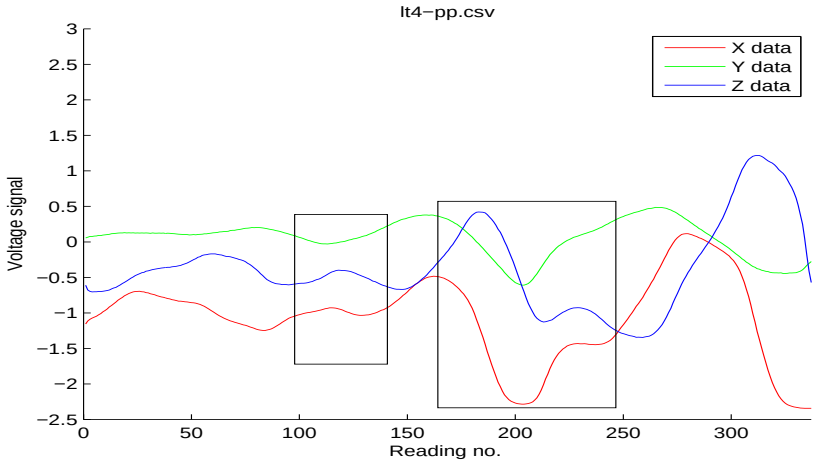


Figure F.14: Data collected when scanning transversely along scan line four after longitudinal energisation with defects 2C and 7BC labelled.

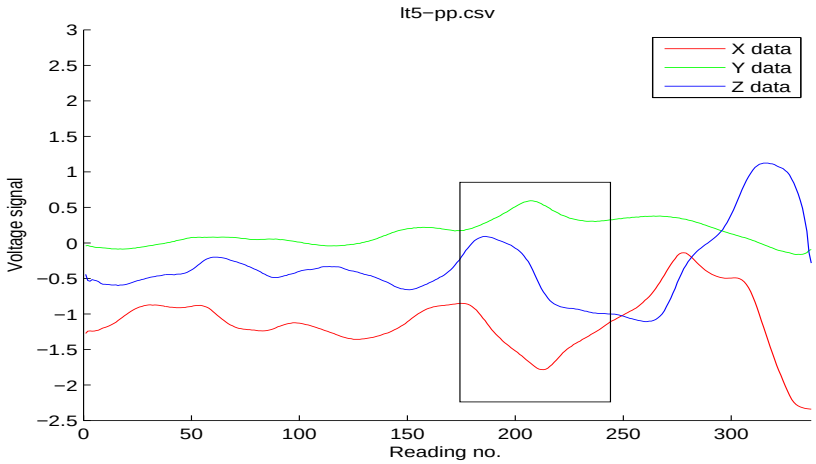


Figure F.15: Data collected when scanning transversely along scan line five after longitudinal energisation with defect 7BC labelled from neighbouring scan line four.

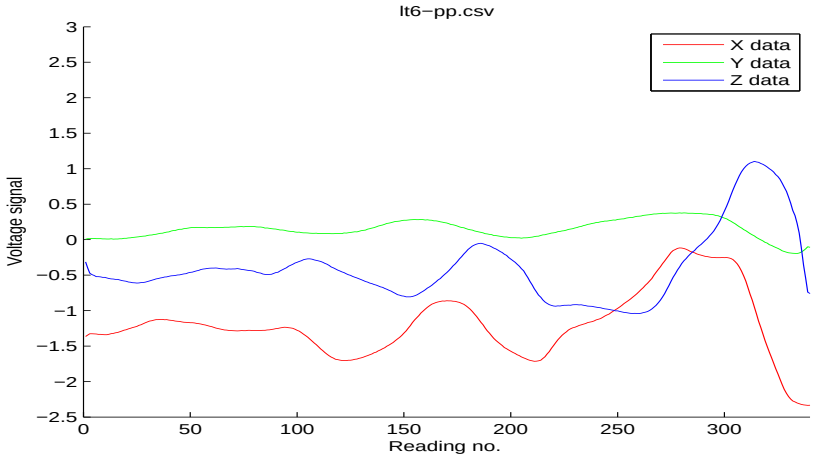


Figure F.16: Data collected when scanning transversely along scan line six after longitudinal energisation.

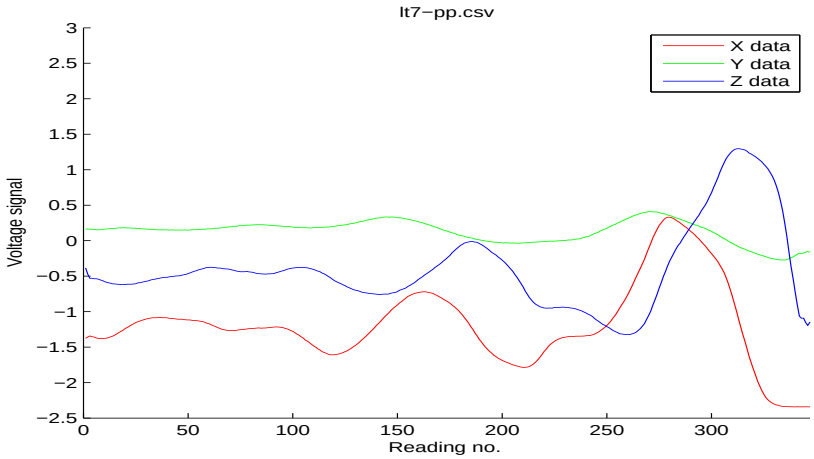


Figure F.17: Data collected when scanning transversely along scan line seven after longitudinal energisation.

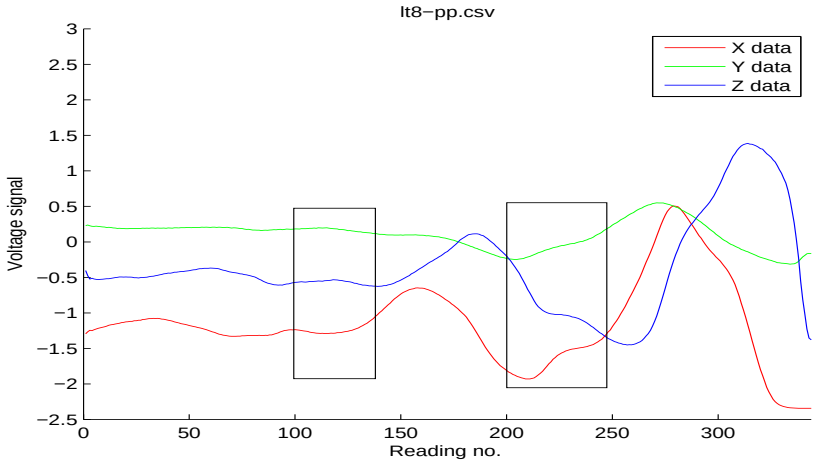


Figure F.18: Data collected when scanning transversely along scan line eight after longitudinal energisation with defects 3C and 8BC labelled.

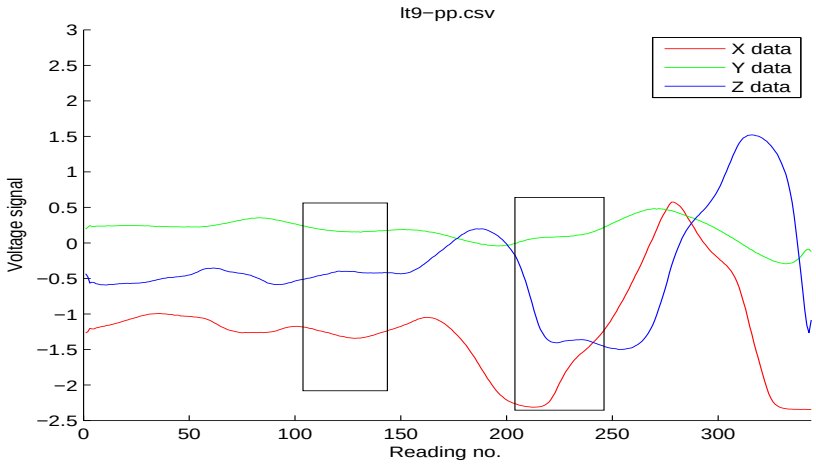


Figure F.19: Data collected when scanning transversely along scan line nine after longitudinal energisation with defects 3C and 8BC labelled.

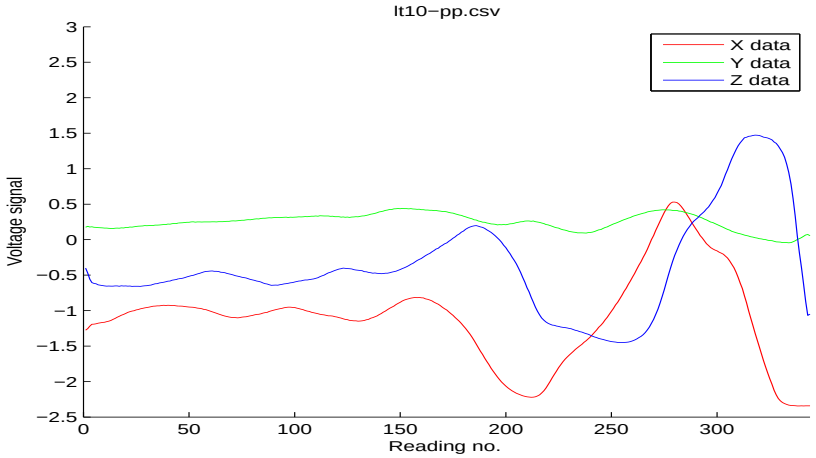


Figure F.20: Data collected when scanning transversely along scan line ten after longitudinal energisation.

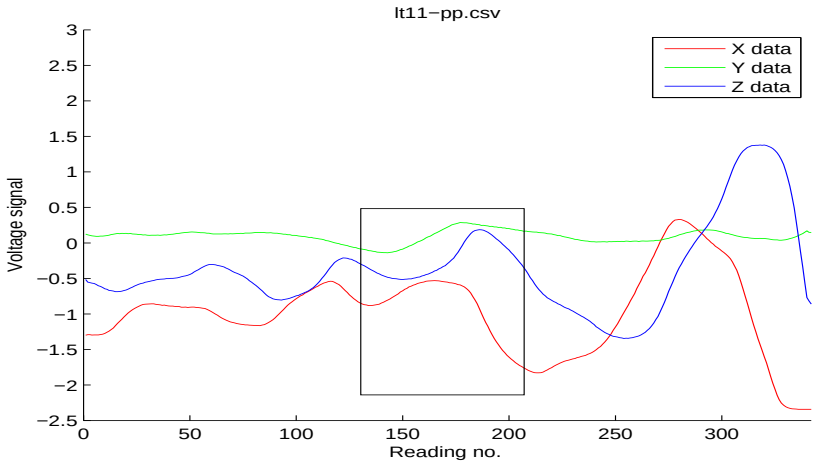


Figure F.21: Data collected when scanning transversely along scan line 11 after longitudinal energisation with defect 4B labelled.

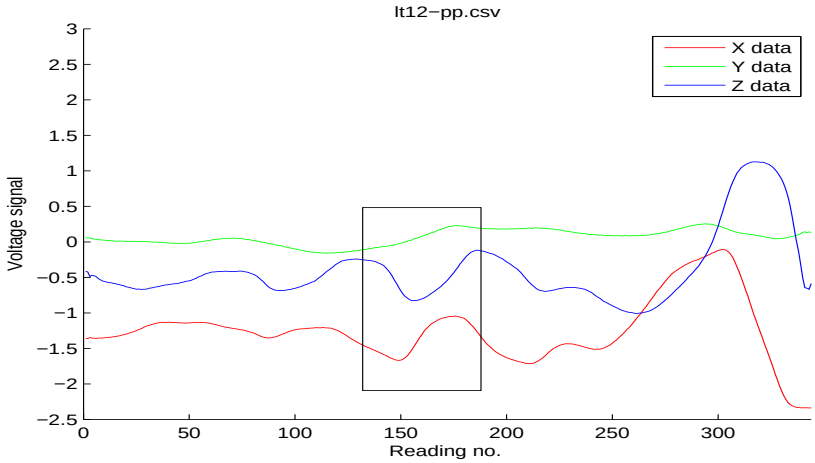


Figure F.22: Data collected when scanning transversely along scan line 12 after longitudinal energisation with defect 4B labelled.

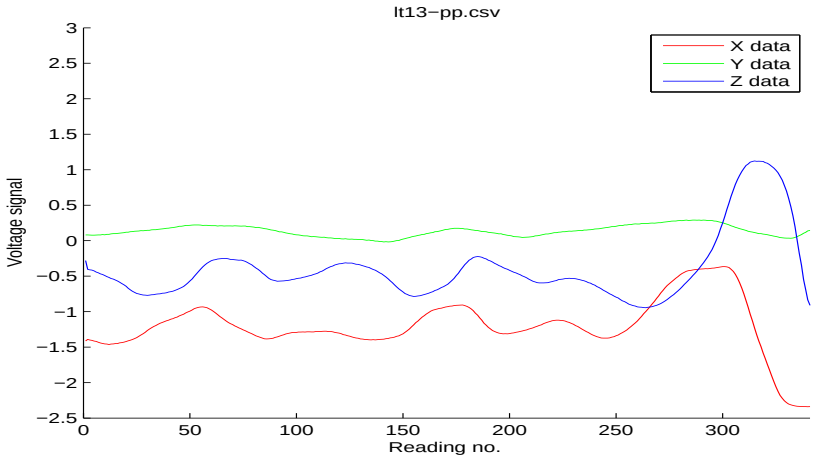


Figure F.23: Data collected when scanning transversely along scan line 13 after longitudinal energisation.

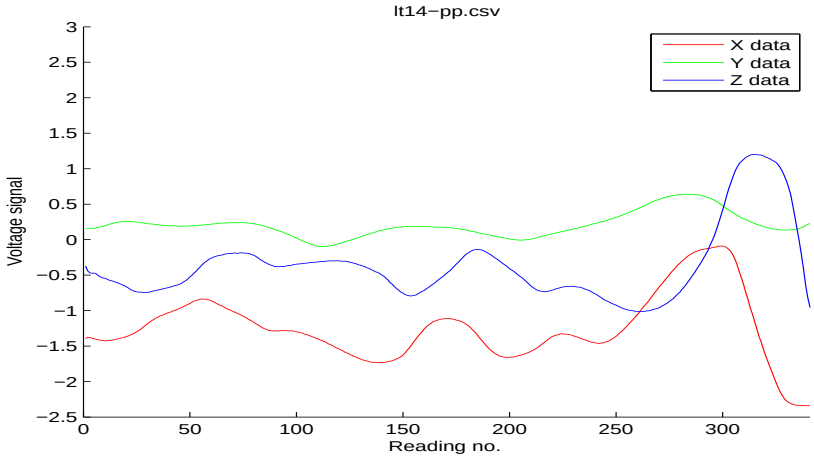


Figure F.24: Data collected when scanning transversely along scan line 14 after longitudinal energisation.

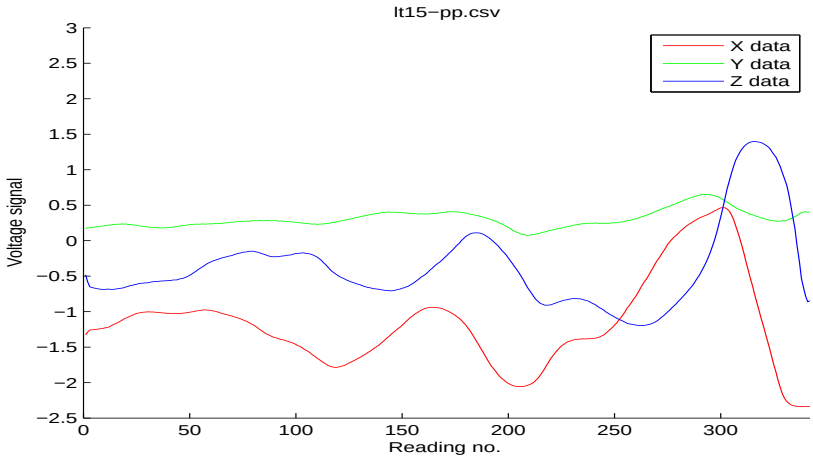


Figure F.25: Data collected when scanning transversely along scan line 15 after longitudinal energisation.

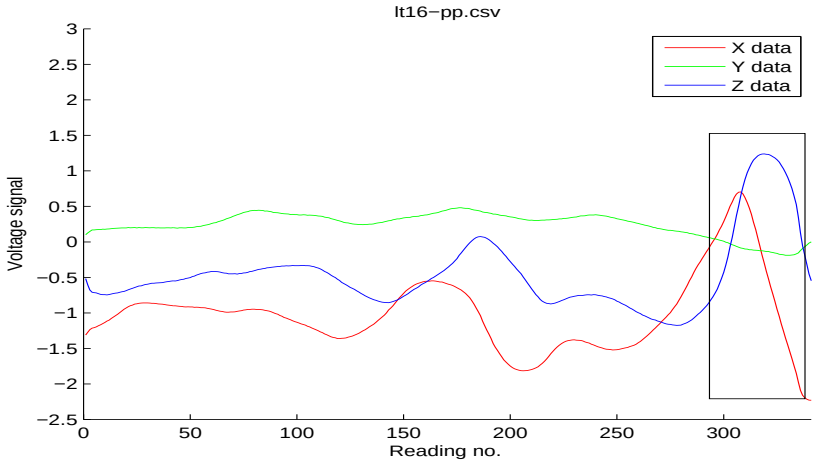


Figure F.26: Data collected when scanning transversely along scan line 16 after longitudinal energisation with defect 11BC labelled.

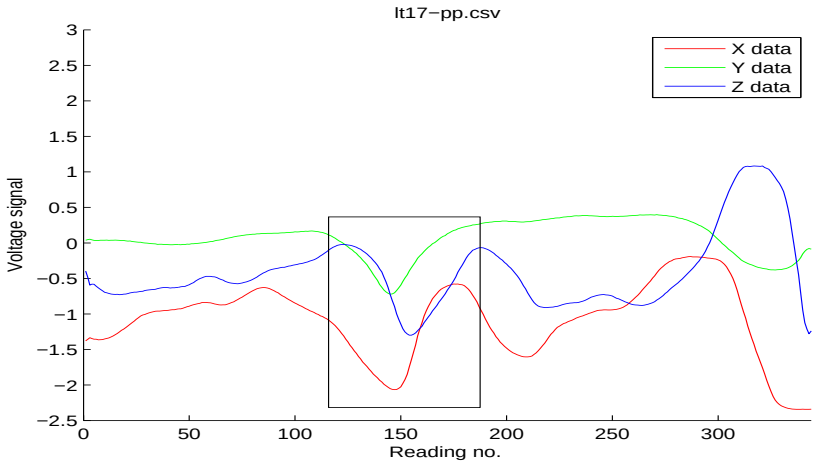


Figure F.27: Data collected when scanning transversely along scan line 17 after longitudinal energisation with defect 5B labelled.

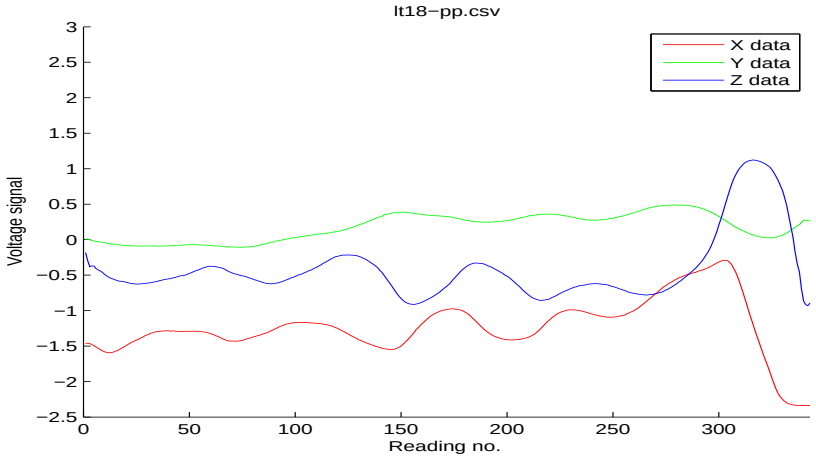


Figure F.28: Data collected when scanning transversely along scan line 18 after longitudinal energisation.

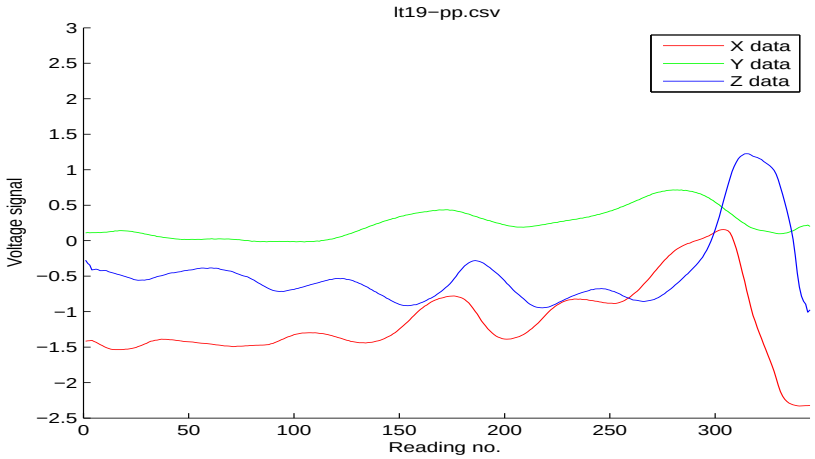


Figure F.29: Data collected when scanning transversely along scan line 19 after longitudinal energisation.

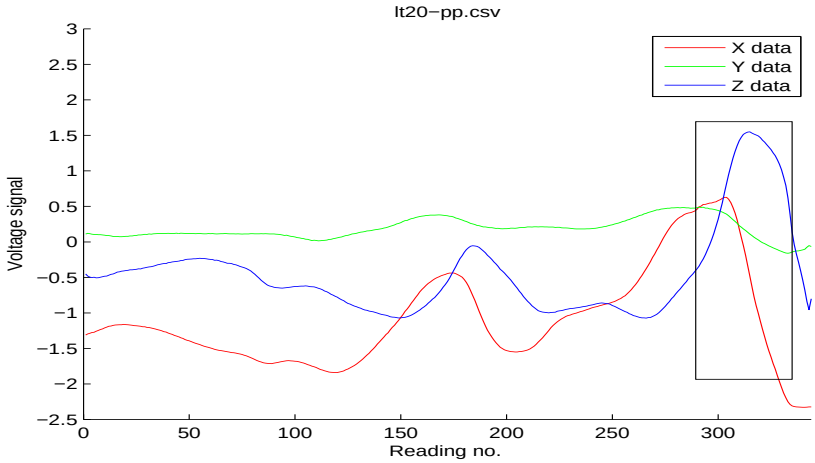


Figure F.30: Data collected when scanning transversely along scan line 20 after longitudinal energisation with defect 12B labelled.

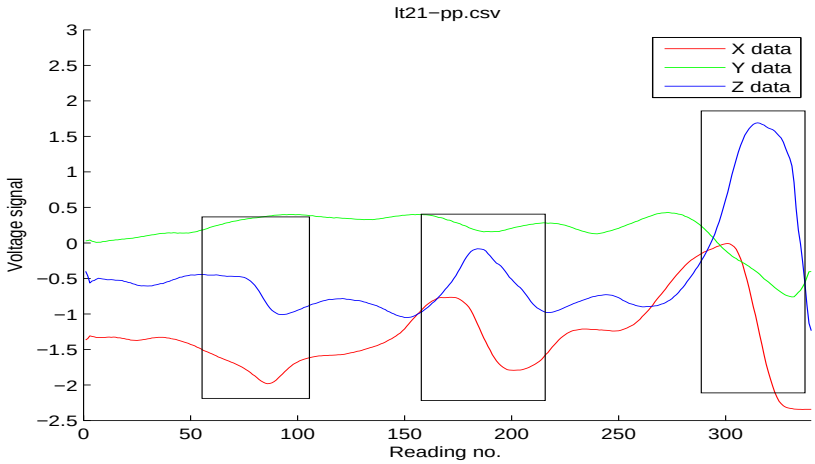


Figure F.31: Data collected when scanning transversely along scan line 21 after longitudinal energisation with defects 1C, 6BC and 12B labelled.

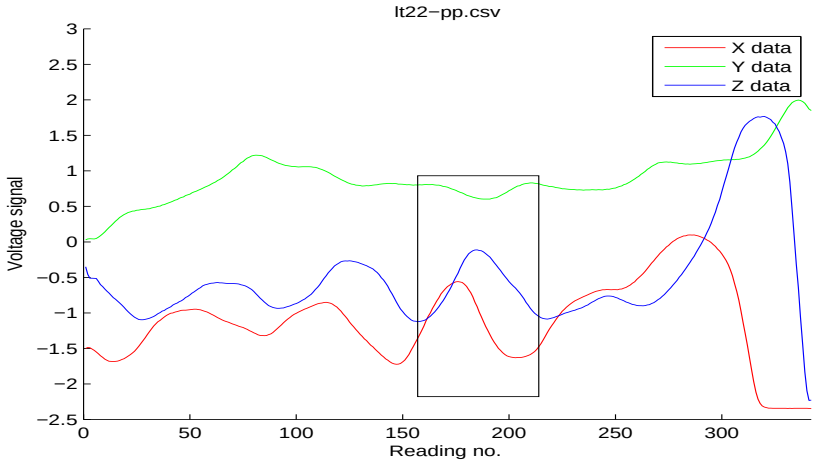


Figure F.32: Data collected when scanning transversely along scan line 22 after longitudinal energisation with defect 6BC labelled.

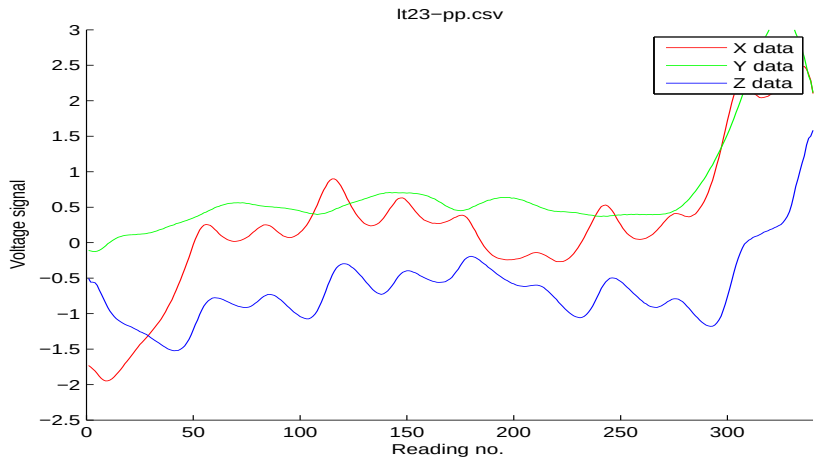


Figure F.33: Data collected when scanning transversely along scan line 23 after longitudinal energisation.

Inspection of the figures in this section shows that energising and scanning longitudinally give the clearest signatures for each defect. Analysis of the transverse scans reveals that the defect signatures are less clear as a result of the influence of the geometry of the structure in relation to the energisation direction. As was discussed in Chapter 5, a large ratio between the length of the energisation direction and the other dimensions of the concrete (width and height) leads to the creation of smaller fringing fields after energisation. Energisation in the transverse direction leads to the creation of larger fringing fields as the mesh is shorter in this direction when compared to its longitudinal length. As a result, energising longitudinally gives a higher signal-to-noise ratio when compared to data which has been collected after energisation transversely.

Appendix G

Further analysis of all defects pre-concrete encasement

This section details further analysis performed to assess the signatures of the corrosion defects introduced following the procedure, outlined in Appendix D, that were difficult to detect when analysing all of the scan lines at once. Figures G.1, G.3 and G.5 show the data collected from the mesh prior to corrosion creation. Figures G.2, G.4 and G.6 show the same scan lines after corrosion had been created on the relevant areas in the mesh.

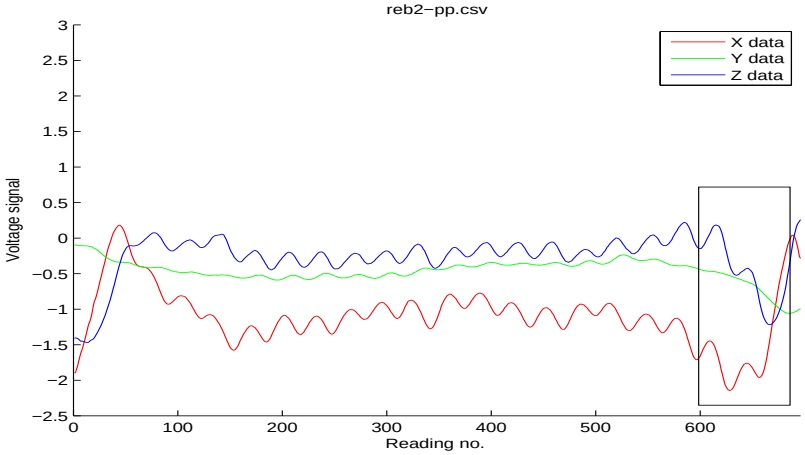


Figure G.1: A scan of longitudinal rebar two prior to corrosion creation with the expected location of defect 1C shown.

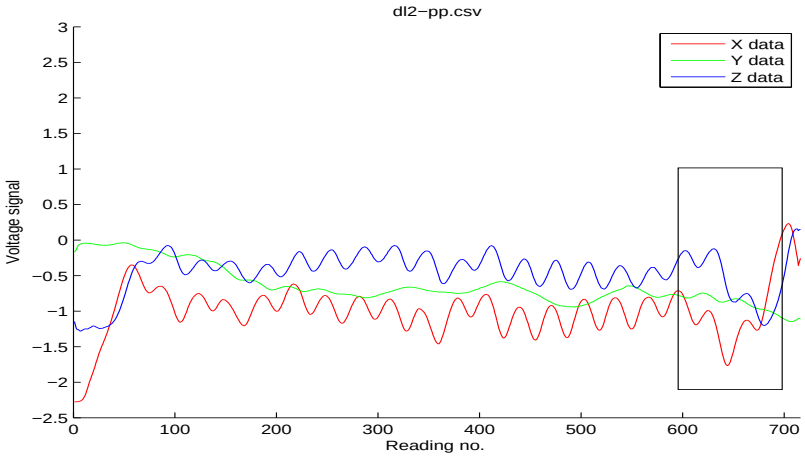


Figure G.2: A scan of longitudinal rebar two after corrosion creation with the location of defect 1C shown.

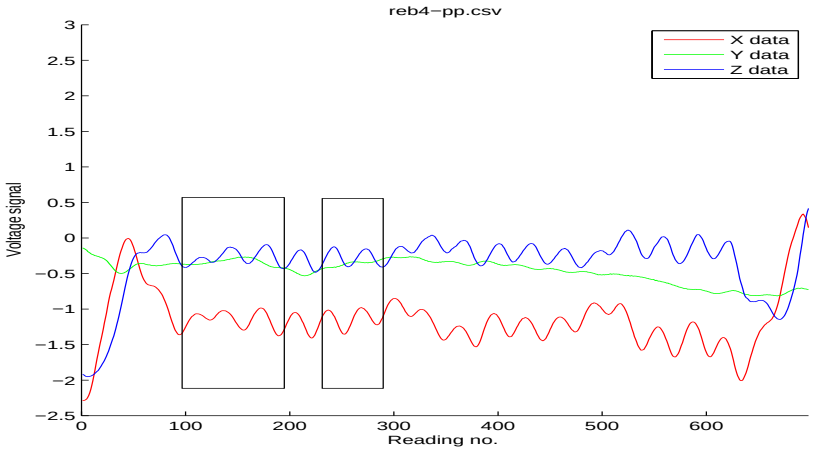


Figure G.3: A scan of longitudinal rebar four prior to corrosion creation with the expected locations of defect 2C and 3C shown.

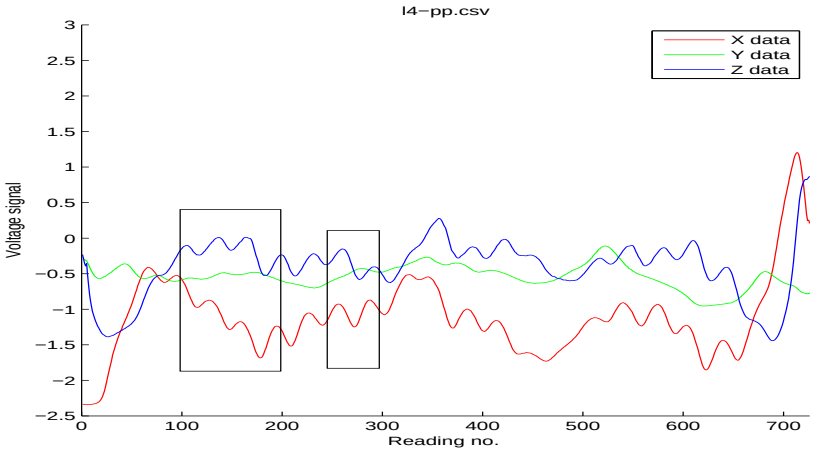


Figure G.4: A scan of longitudinal rebar four after corrosion creation with the locations of defects 2C and 3C shown.

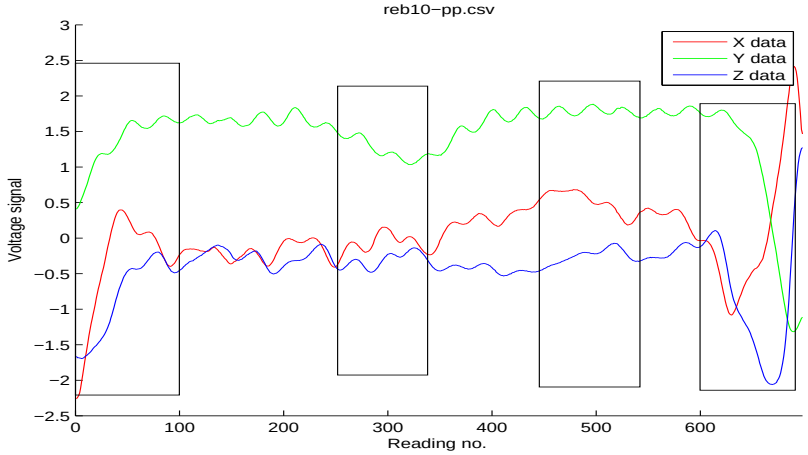


Figure G.5: A scan of longitudinal rebar ten prior to corrosion and break creation with the expected locations of defects 9B, 10C, 11BC and 12B highlighted.

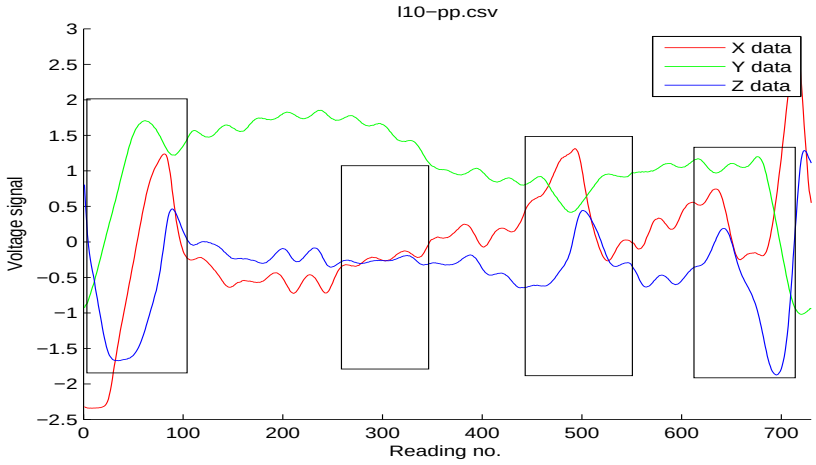


Figure G.6: A scan of longitudinal rebar 10 after corrosion creation.

Comparison of Figures G.1, G.2, G.3, G.4, G.5 and G.6 show that it is difficult to see any of the corrosion defects by inspection. Upon closer analysis, however, small signatures can be seen. Analysis of Figures G.1 and G.2 reveals a small signature for

defect 1C around the 667^{th} datapoint. Here the X curve in the post-corrosion scan has a much smaller trough towards the end of the scan line, whereas the pre-corrosion creation has a typical end effect curve. The Z curve is also different in the two plots, the pre-corrosion curve having a deep trough followed by a sharp increase due to the end of the mesh. The post-corrosion curve, however, has no trough: instead it increases drastically again, then levels out, and where the pre-corrosion plot would turn into a trough again, post corrosion it increases to give a typical end effect curve. This could, therefore, be the characteristic signature for defect 1C.

Figures G.3 and G.4 show data collected from energising and scanning longitudinally along rebar four where defects 2C and 3C were created. These defects should be located approximately around the 127^{th} and 270^{th} datapoints respectively. The plots in Figure G.3 show a periodic pattern in the X and Z direction where one almost mirrors the other. In Figure G.4, however, this periodicity is less prominent, particularly between the 100^{th} and 200^{th} datapoints and between the 250^{th} and 300^{th} datapoints. These two ranges correlate to defects 2C and 3C. There is also an interesting peak shortly after the signature of defect 3C between the 300^{th} and 400^{th} datapoints. This is probably from defect 4B which was created on the neighbouring rebar five. From datapoint 500 onwards, the post-corrosion data returns to a periodicity similar to that seen in the pre-corrosion data.

From inspection of Figures G.5 and G.6, one can see clear differences in the pre and post-defect introduction data. For example, the first 100 datapoints in both figures are very different; Figure G.5 has a small change in signal which is typically seen at the start of scan lines, whereas Figure G.6 has a large peak over the same 100 datapoints which is due to the break created for defect 9B. Analysis of both Figures around the 300^{th}

datapoint also shows some difference, albeit a lot smaller than the previously discussed defect 9B. The data around this area in Figure G.6 can therefore be interpreted as the corrosion created for defect 10C. In conclusion, all of the defects containing rust alone (defects 1C, 2C, 3C and 10C) can be seen after heating of the mesh as described earlier in Appendix D.

Corrosion analysis of the data collected from energising and scanning the concrete longitudinally after defect creation has revealed that 11 out of the 12 defects could be observed. Figures G.7 and G.8 show data collected between rebars five and six where defect 6BC should be present around the 698^{th} datapoint. As the Figures show, both plots are quite similar towards the end of the scan line due to the presence of end effects, but some difference in the end effect of Figure G.8 exists. Despite this small difference between the two plots, the end effect signal still masks the majority of the signature of defect 6BC. In order to find the signature for defect 6BC, data collected when scanning the mesh transversely was analysed. Figures G.9 and G.10 show the data collected from scanning transversely along rebar 22 on which defect 6BC is located.

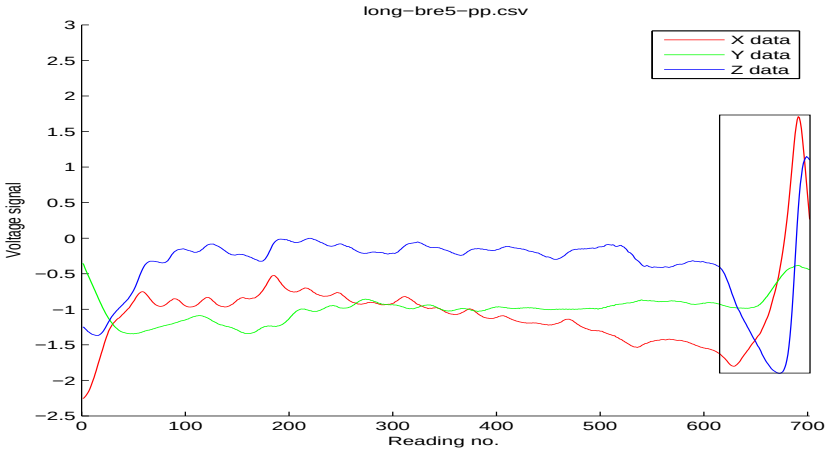


Figure G.7: A scan between longitudinal rebar five and six before corrosion and break creation showing the expected location of defect 6BC.

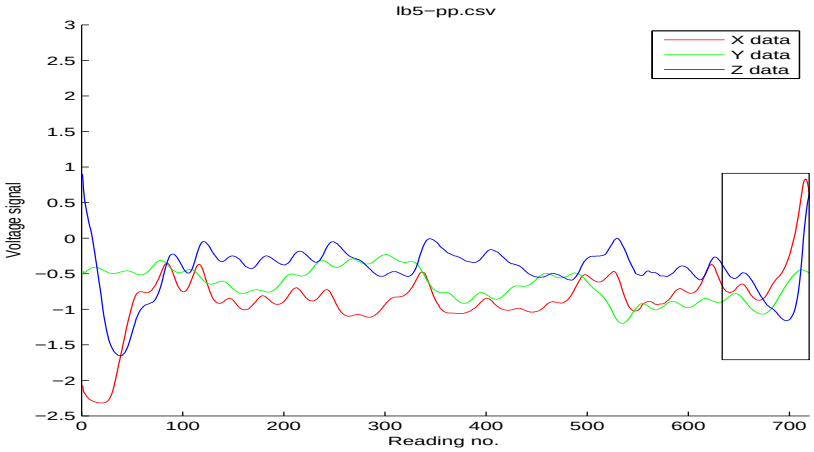


Figure G.8: A scan between longitudinal rebar five and six after corrosion and break creation showing the location of defect 6BC.

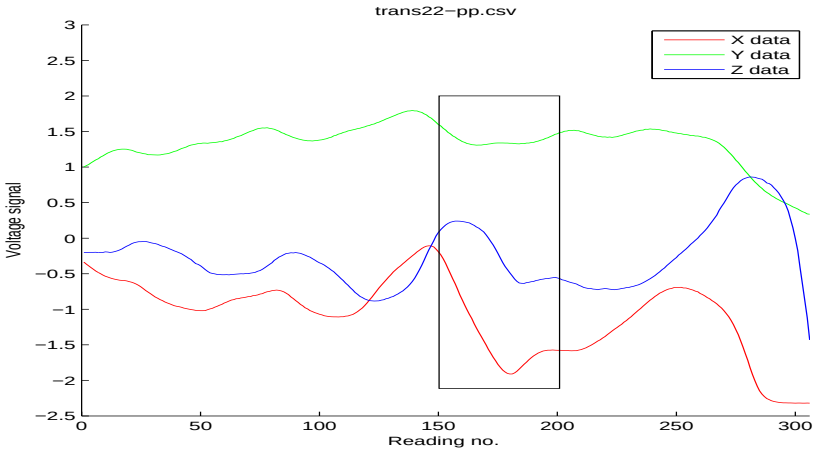


Figure G.9: A scan of transverse rebar 22 prior to break and corrosion insertion showing the expected location of defect 6BC.

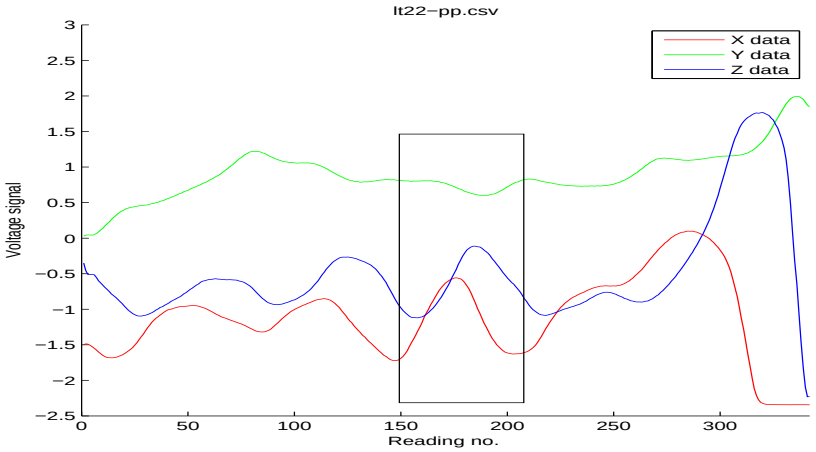


Figure G.10: A scan of transverse rebar 22 after break and corrosion insertion showing the location of defect 6BC.

Analysis of Figures G.9 and G.10 shows a difference around the 175^{th} datapoint which is where defect 6BC should approximately occur. Therefore, defect 6BC can be

seen when energising the mesh longitudinally and scanning it transversely, rather than scanning longitudinally. This analysis revealed that all of the 12 defects, including the smaller corrosion defects, could be detected prior to concrete encasement.

Appendix H

Data collected by energising and scanning each individual scan line

The figures below show all ten scan lines collected when energising and scanning each individual one before moving onto the next scan line which was then energised and scanned. This differs from the standard approach used in the rest of this work where all scan lines were energised as a whole, and then scanned afterwards.

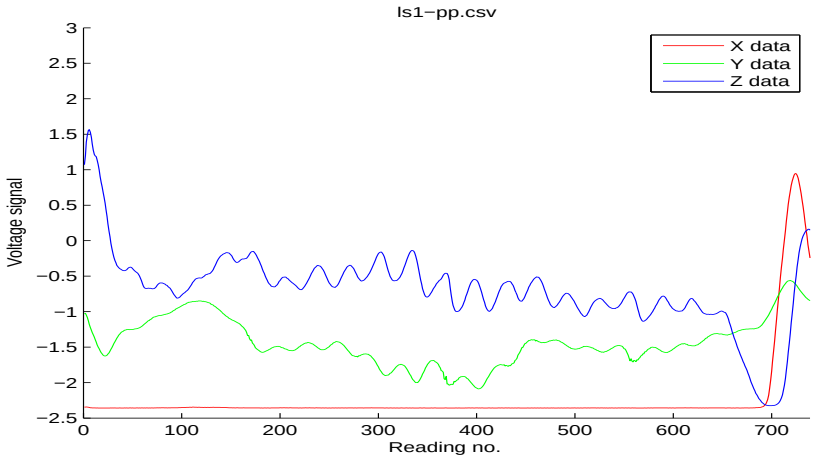


Figure H.1: Data collected from scan line one when energising and scanning the individual scan line.

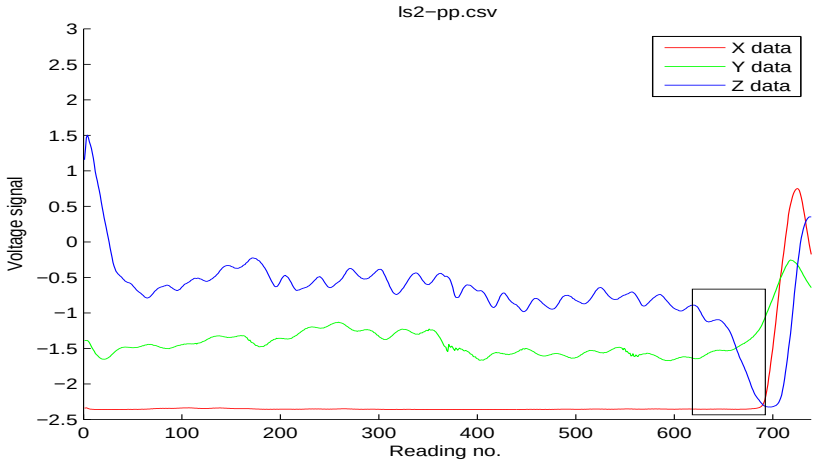


Figure H.2: Data collected from scan line two when energising and scanning the individual scan line with the location of defect 1C shown.

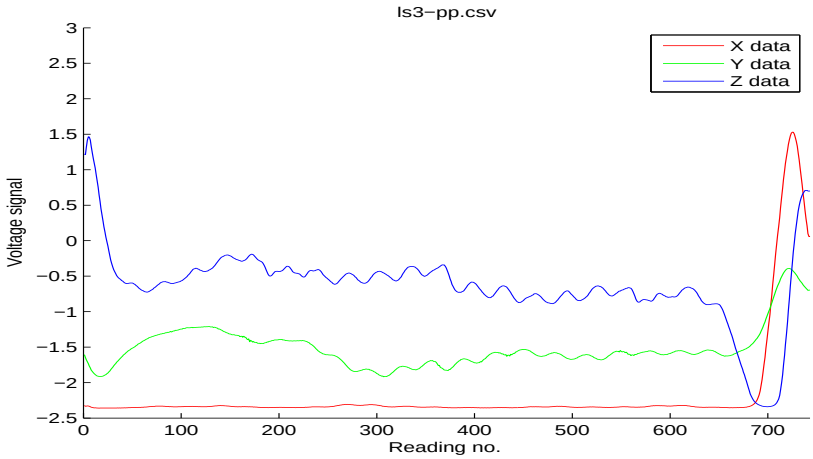


Figure H.3: Data collected from scan line three when energising and scanning the individual scan line.

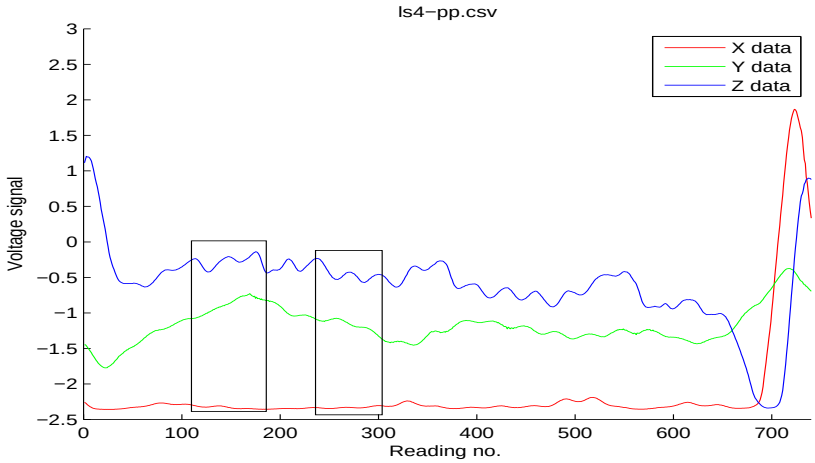


Figure H.4: Data collected from scan line four when energising and scanning the individual scan line with the locations of defect 2C and 3C shown.

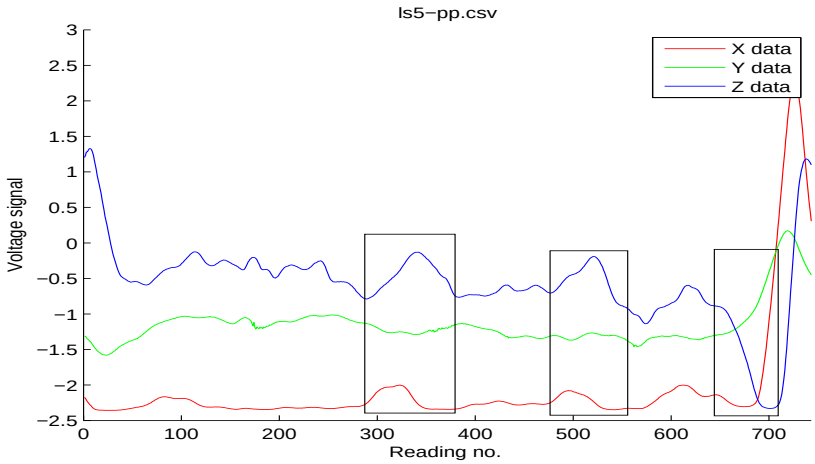


Figure H.5: Data collected from scan line five when energising and scanning the individual scan line with the locations of defect 4B, 5B and 6BC shown.

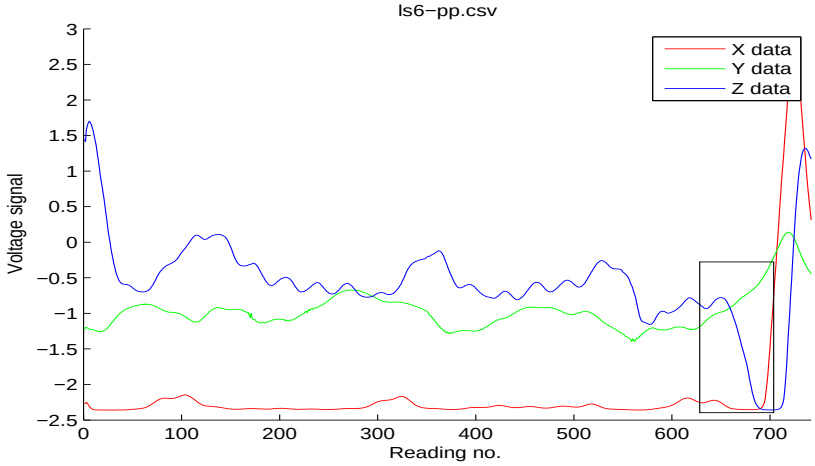


Figure H.6: Data collected from scan line six when energising and scanning the individual scan line with the location of defect 6BC shown.

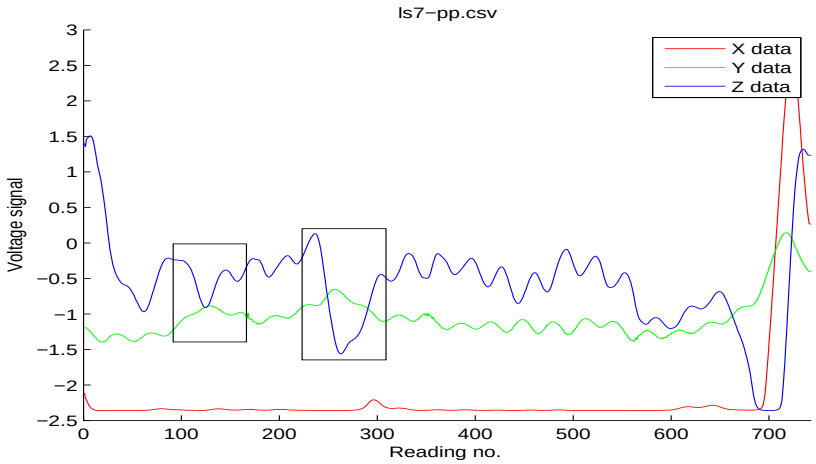


Figure H.7: Data collected from scan line seven when energising and scanning the individual scan line with the locations of defect 7BC and 8BC shown.

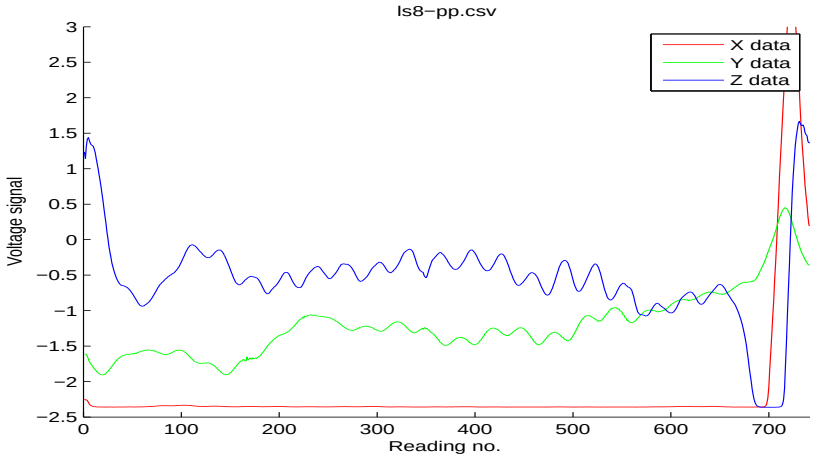


Figure H.8: Data collected from scan line eight when energising and scanning the individual scan line.

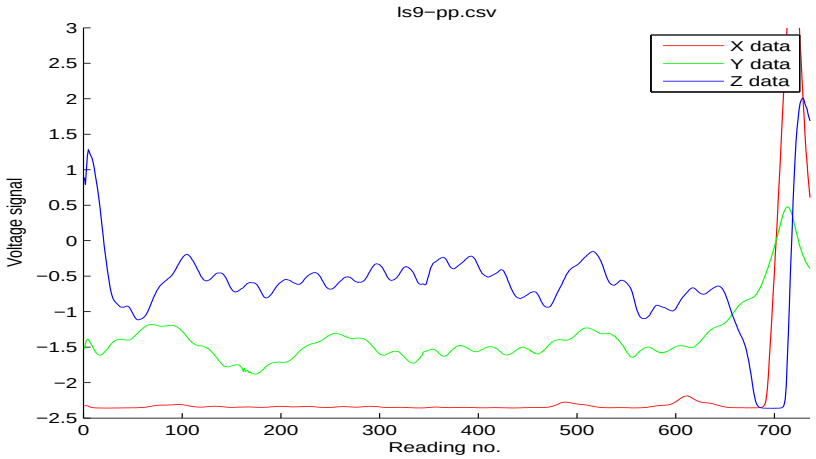


Figure H.9: Data collected from scan line nine when energising and scanning the individual scan line.

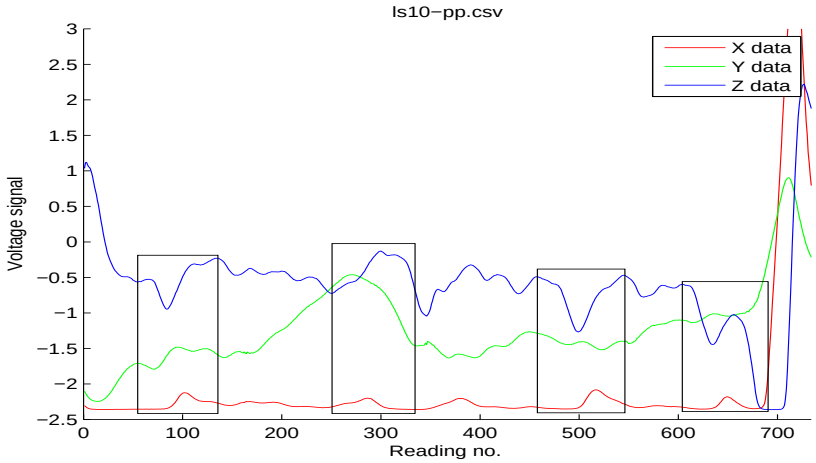


Figure H.10: Data collected from scan line ten when energising and scanning the individual scan line where the locations of defects 9B, 10C, 11BC and 12B are shown.

The data presented in this section shows that as a result of energising and scanning each individual rebar before repeating the process on the next rebar increases the

fringing field strength of the rebar currently under investigation. This was expected due to the fact that energisation was not applied to the next rebar (as is the case when the whole mesh is energised and scanned afterwards), which usually dissipates the fringing field of the previously applied energisation. However, this also increased the strength of the overall signal received, as more information about irrelevant characteristics of the mesh, such as the location of the healthy rebars, was now present in the data collected. Therefore, energising the whole structure prior to scanning increases the SNR ratio of the areas of the mesh that are of most interest.

Appendix I

Data collected using reverse energisation direction

The figures in this section show the data collected when applying the energiser first to scan line ten and finishing at scan line one. This application of the energisation is the opposite of the procedure that was originally used in this research. The location of each defect is also shown on the scan line where it occurred, as well as in neighbouring scan lines where its signature was detectable.

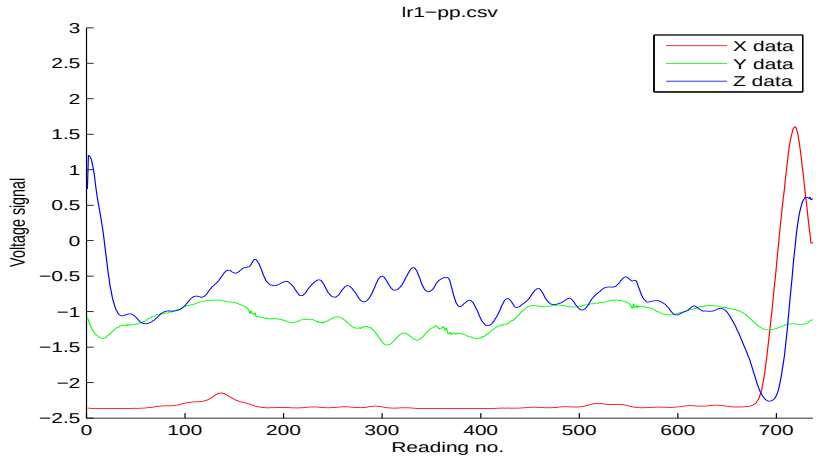


Figure I.1: Data collected from scan line one when energising in the reverse direction (starting at scan ten and finishing at scan one).

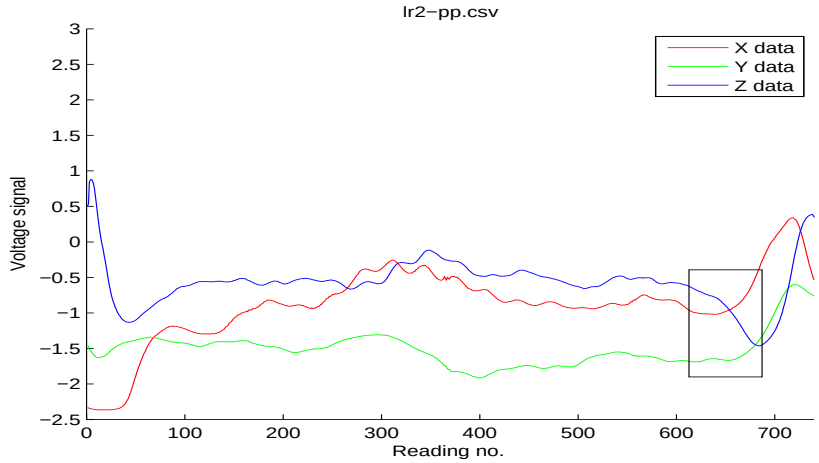


Figure I.2: Data collected from scan line two when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defect 1C shown.

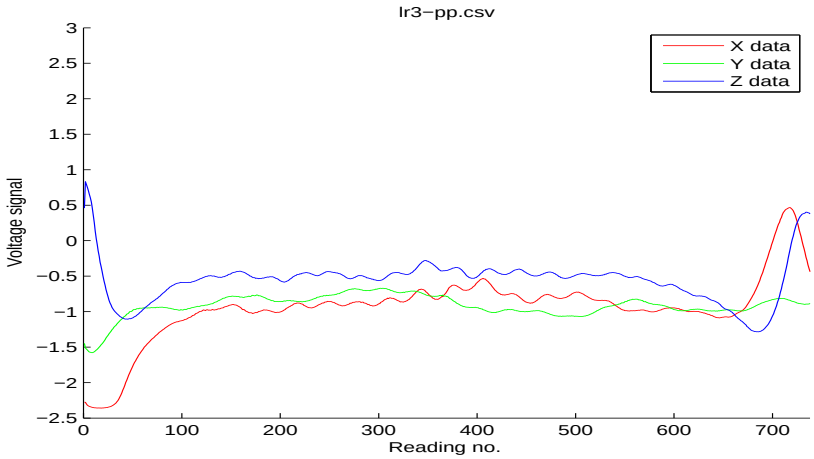


Figure I.3: Data collected from scan line three when energising in the reverse direction (starting at scan ten and finishing at scan one).

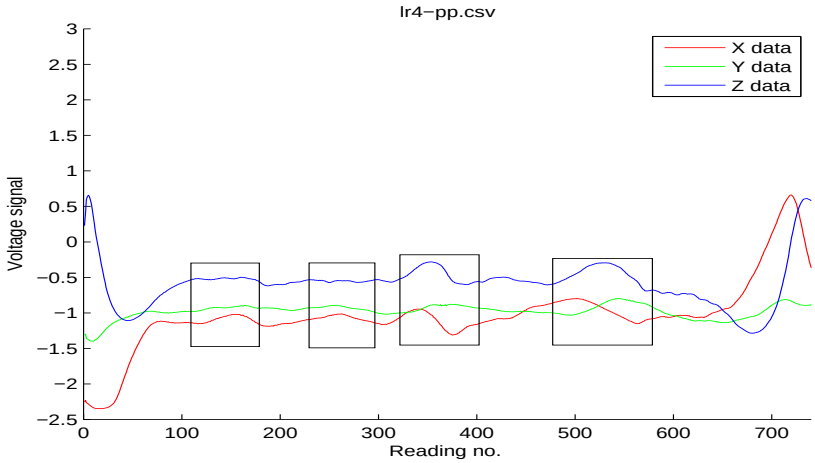


Figure I.4: Data collected from scan line four when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defects 2C and 3C shown, as well as the signatures of defects 4B and 5B from neighbouring scan line five.

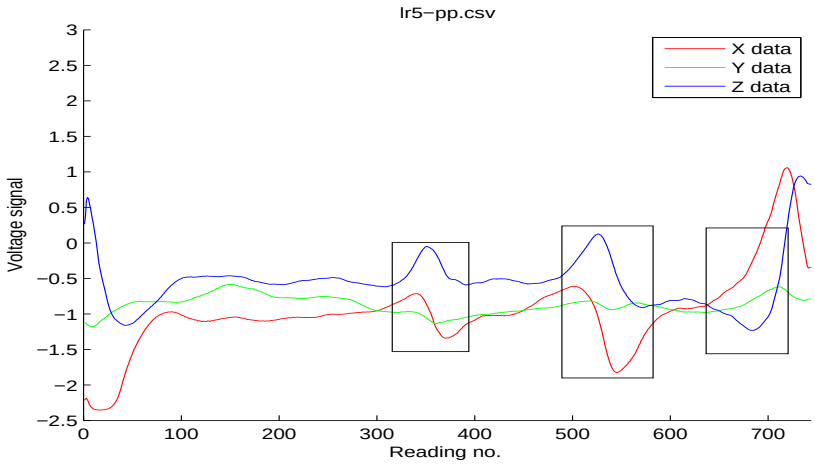


Figure I.5: Data collected from scan line five when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defects 4B, 5B and 6BC shown.

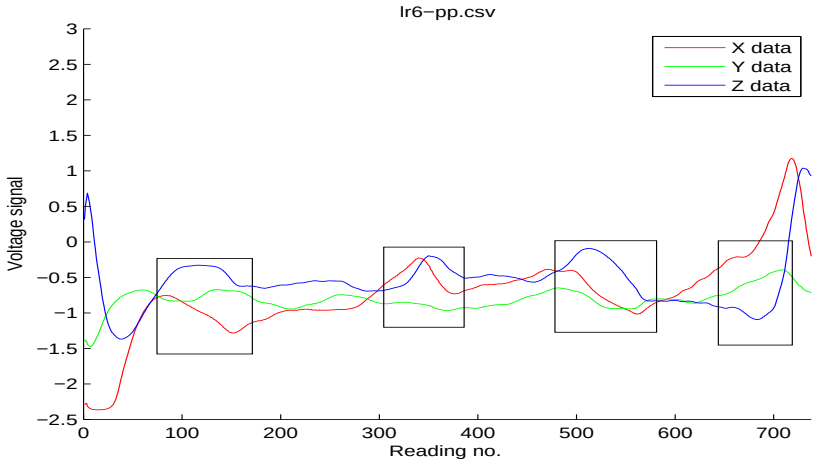


Figure I.6: Data collected from scan line six when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defect 6BC shown, as well as the signatures of defects 7BC, 4B and 5B.

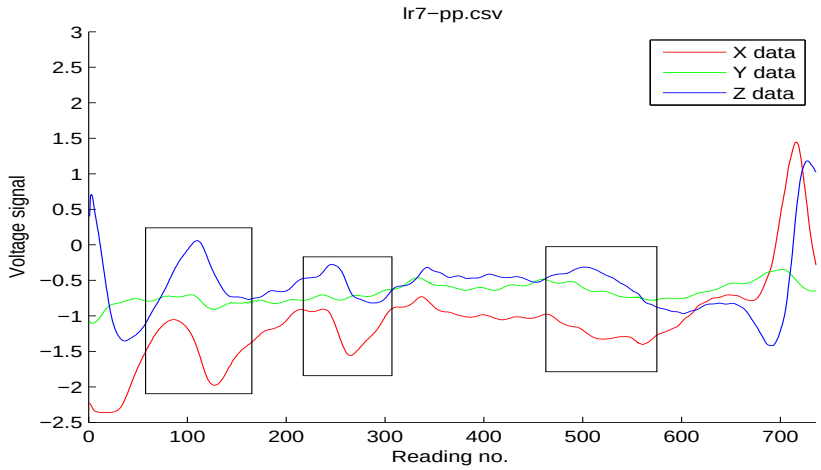


Figure I.7: Data collected from scan line seven when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defects 7BC and 8BC shown, as well as a faint signature of defect 5B from scan line five.

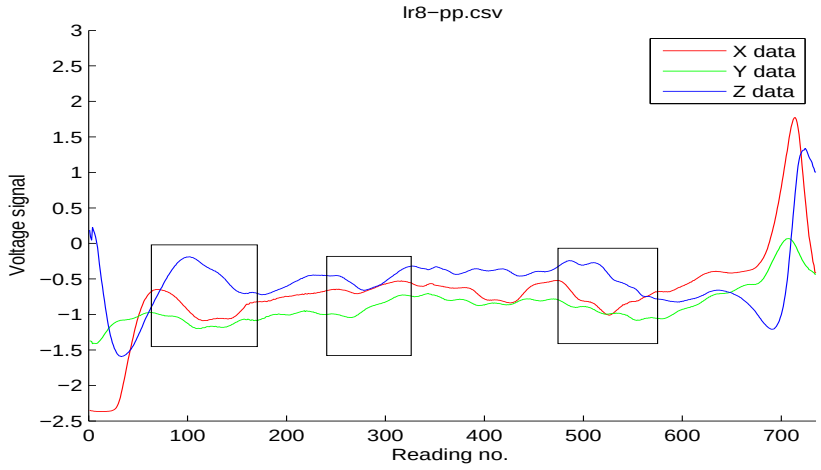


Figure I.8: Data collected from scan line eight when energising in the reverse direction (starting at scan ten and finishing at scan one) with the signatures of defects 7BC, 8BC and 5B shown.

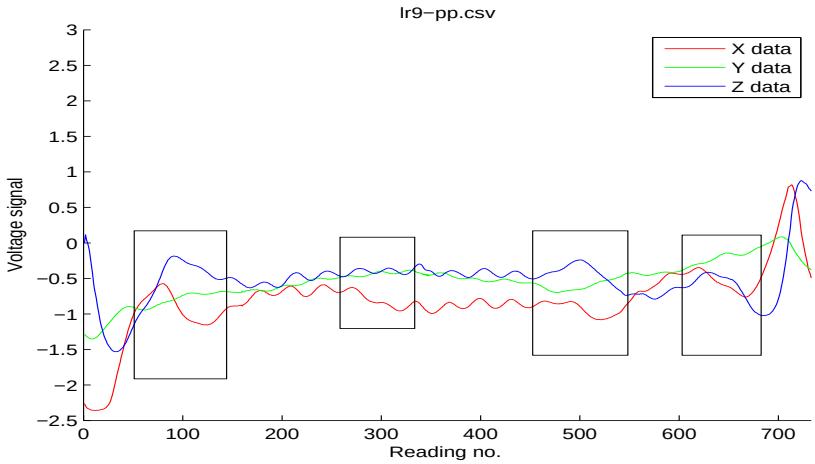


Figure I.9: Data collected from scan line nine when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defects 9B, 10C, 11BC and 12B shown.

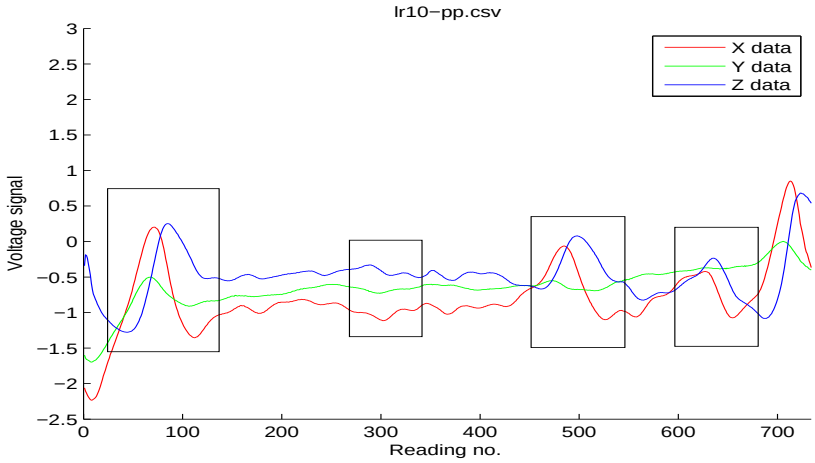


Figure I.10: Data collected from scan line ten when energising in the reverse direction (starting at scan ten and finishing at scan one) with the location of defects 9B, 10C, 11BC and 12B shown.

Analysis of the data collected from scanning longitudinally after energising from scan line ten to one shows that the SNR of the later scan lines increased, especially scans nine and ten where the defect signatures were much clearer as a result of the removal of noise from other features of the mesh including the positions of healthy rebars.

Appendix J

Longitudinal scan lines of all defects post-concrete encasement

Below are the scan lines collected from the first week after the reinforcing mesh was encased in concrete when energising longitudinally and scanning longitudinally.

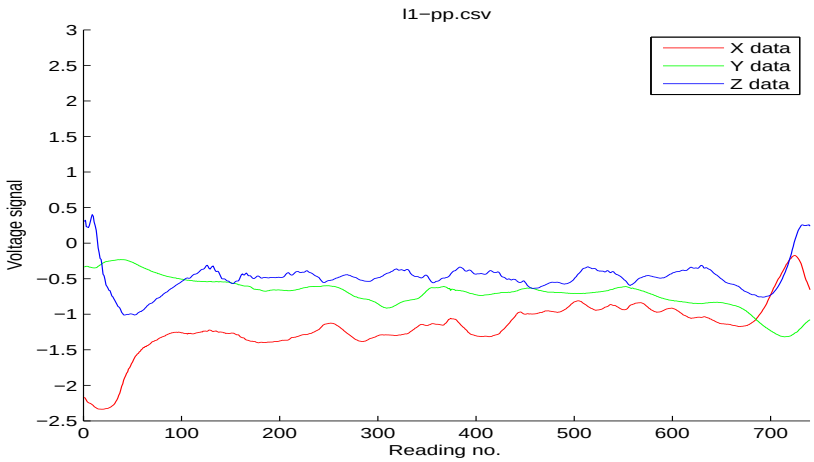


Figure J.1: Data collected from the first week of data collection post concrete encasement showing scan line one when energising and scanning longitudinally.

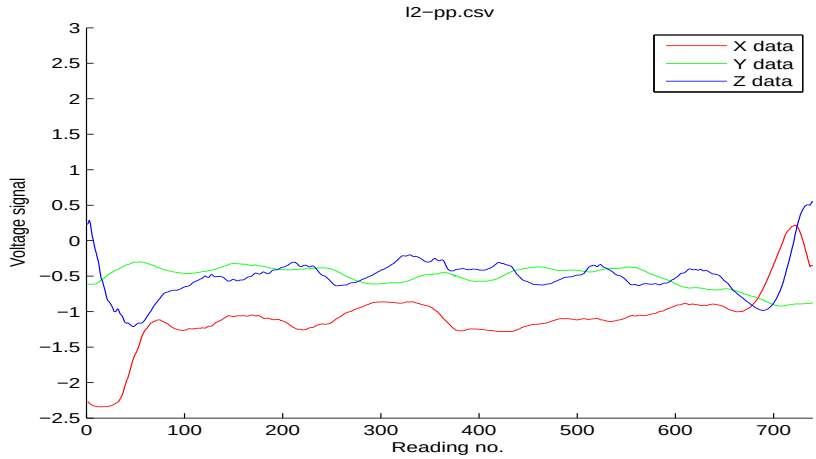


Figure J.2: Data collected from the first week of data collection post concrete encasement showing scan line two when energising and scanning longitudinally. Note that defect 1C which should appear around the 667^{th} data point is no longer visible and hence, is not labelled.

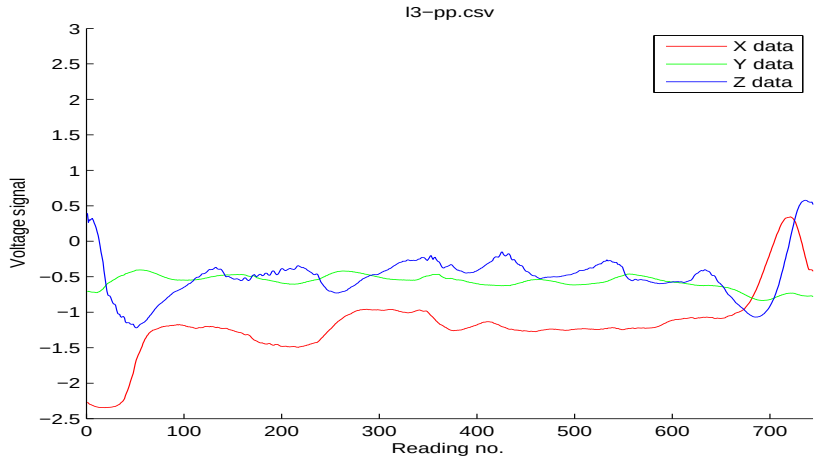


Figure J.3: Data collected from the first week of data collection post concrete encasement showing scan line three when energising and scanning longitudinally. Note that defects 2C and 3C are no longer visible after concrete encasement and hence, are not labelled.

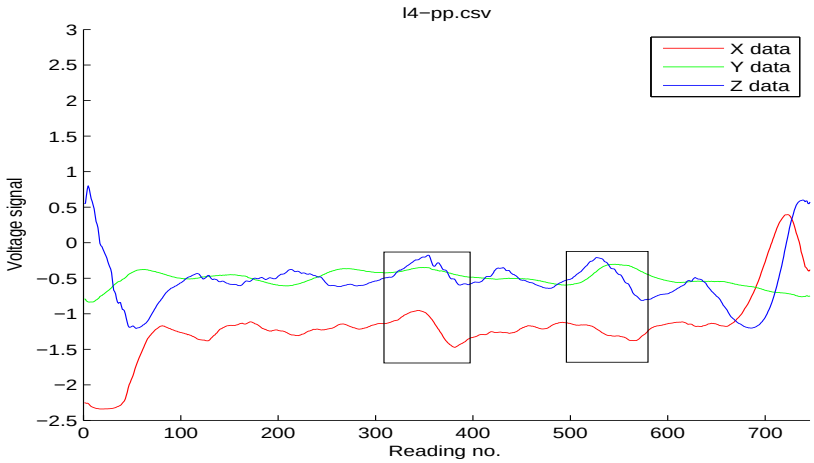


Figure J.4: Data collected from the first week of data collection post concrete encasement showing scan line four when energising and scanning longitudinally. Defects 4B and 5B from neighbouring scan line five are visible and hence are labelled.

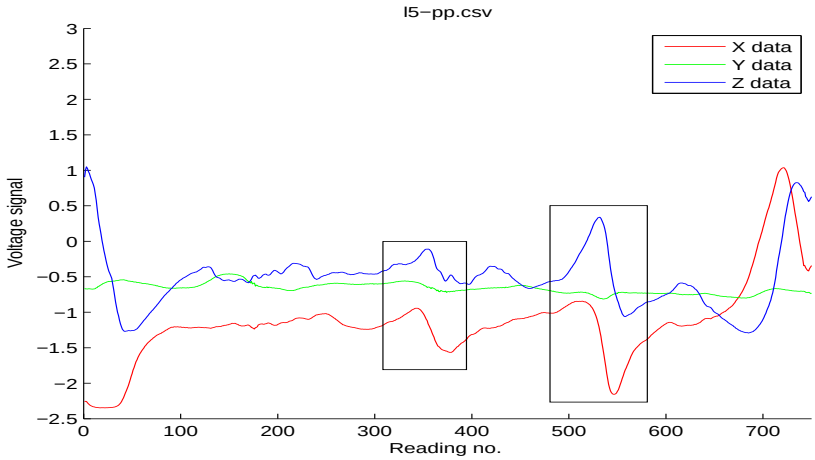


Figure J.5: Data collected from the first week of data collection post concrete encasement showing scan line five when energising and scanning longitudinally. Here defects 4B and 5B are visible and are labelled, whereas defect 6BC is not visible when scanning longitudinally, and hence is not labelled.

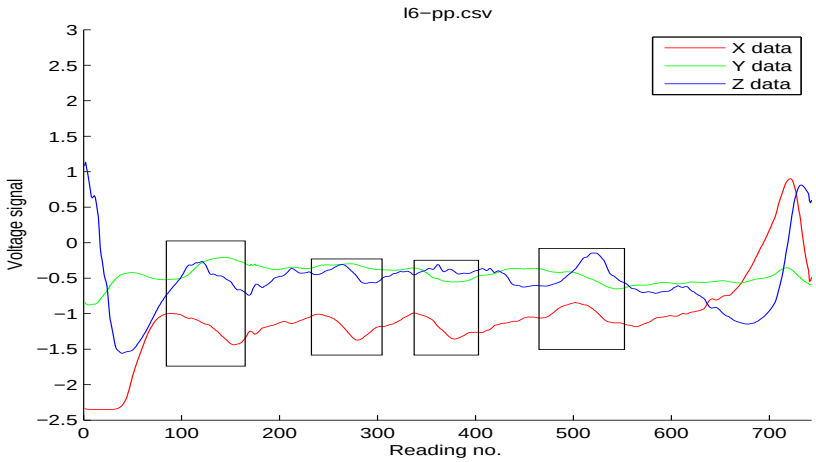


Figure J.6: Data collected from the first week of data collection post concrete encasement showing scan line six when energising and scanning longitudinally. Here defects 4B, 5B, 7BC and 8BC are visible and are highlighted. As with scan line five shown in Figure L.9 defect 6BC is still not visible and is not labelled.

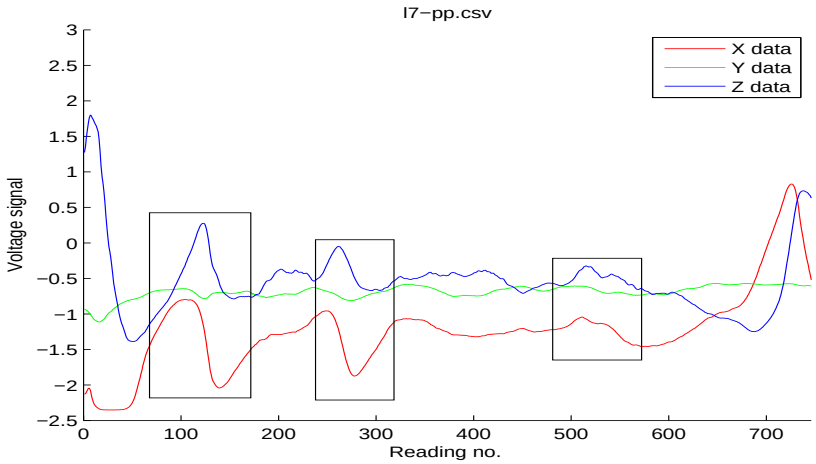


Figure J.7: Data collected from the first week of data collection post concrete encasement showing scan line seven when energising and scanning longitudinally. Here defects 7BC and 8BC are visible and are highlighted, as is a small signature from defect 5B located on rebar five.

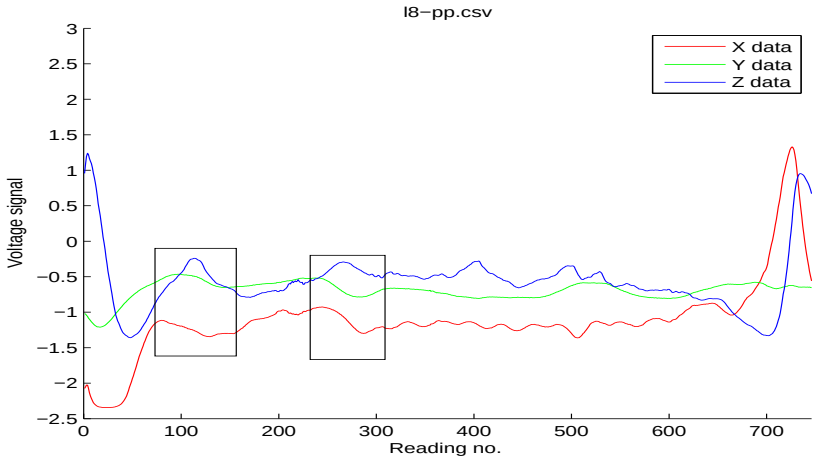


Figure J.8: Data collected from the first week of data collection post concrete encasement showing scan line eight when energising and scanning longitudinally. Here defects 7BC and 8BC from scan line seven are visible and are highlighted.

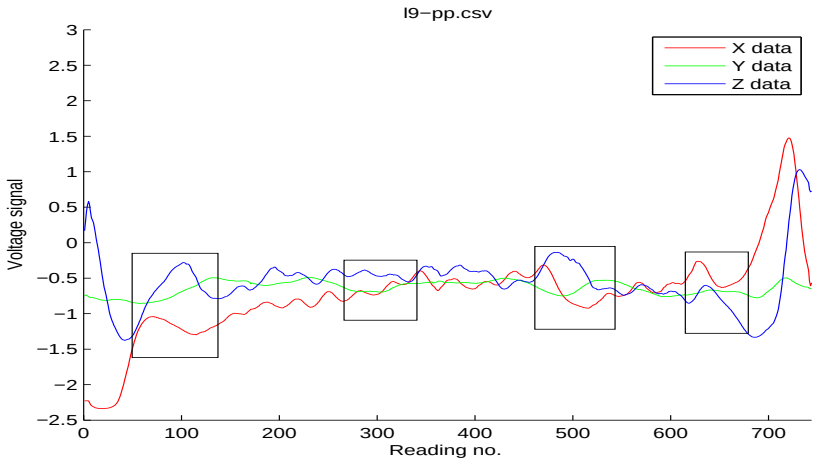


Figure J.9: Data collected from the first week of data collection post concrete encasement showing scan line nine when energising and scanning longitudinally. Here defects 9B, 10C, 11BC and 12B are visible and are highlighted.

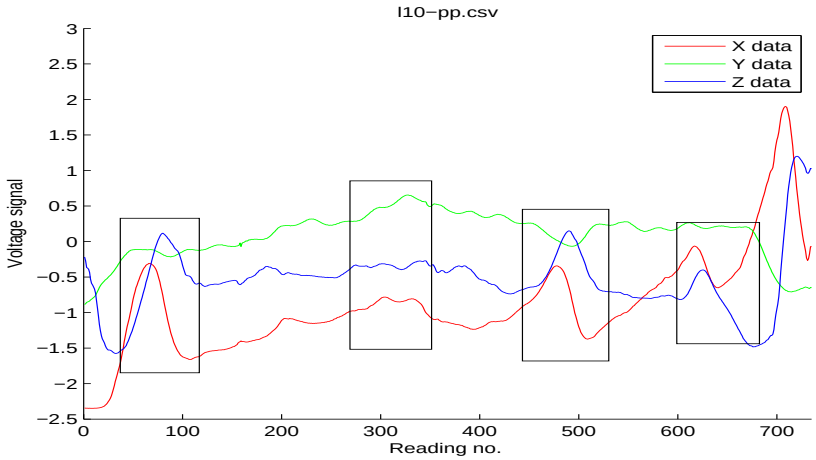


Figure J.10: Data collected from the first week of data collection post concrete encasement showing scan line ten when energising and scanning longitudinally. Here defects 9B, 10C, 11BC and 12B are visible and are highlighted.

Appendix K

Analysis of corrosion defects post-concrete encasement

After concrete encasement, the corrosion defects introduced into the mesh were difficult to detect when looking at the scan lines as a whole. This section analyses each scan line in more detail in order to detect the signatures of the corrosion defects. Figures K.1, K.2 and K.3 show the EMAD data collected from scanning scan lines two, four and ten where defects 1C, 2C, 3C and 10C are located.

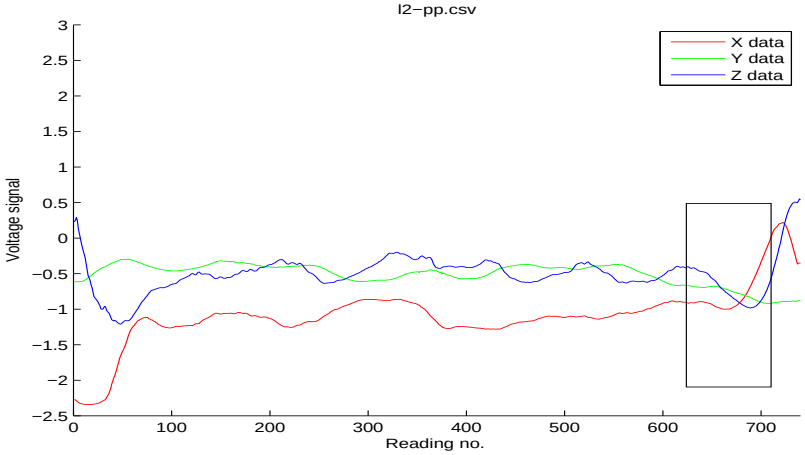


Figure K.1: Data collected from scan line two after concrete encasement where defect 1C is located as shown by the black box. As a result of encasement the signature of this defect is no longer visible.

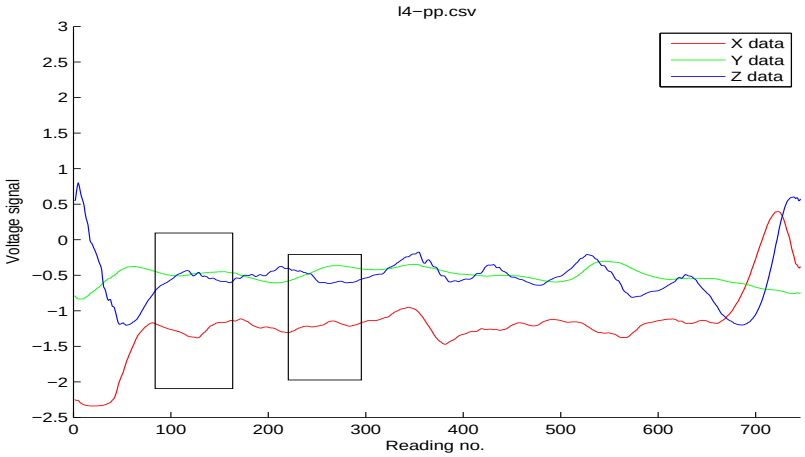


Figure K.2: Data collected from scan line four after concrete encasement where defects 2C and 3C are located as shown by the two boxes. As with defect 1C it is difficult to spot these two defect's signatures from this data.

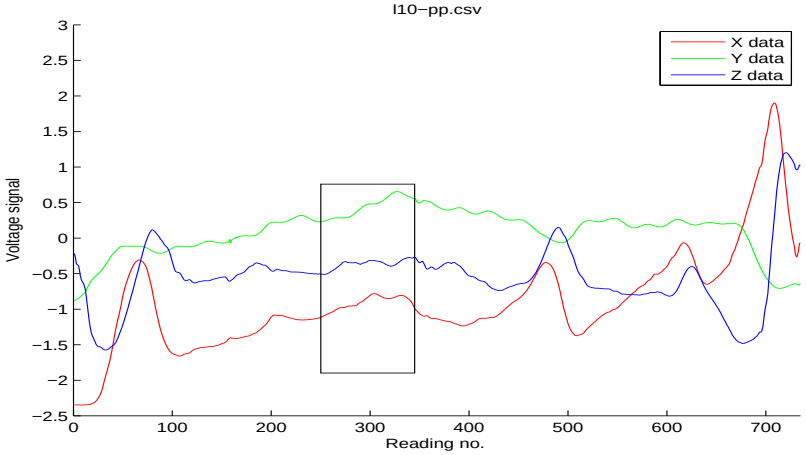


Figure K.3: Data collected from scan line ten after concrete encasement where (among others) defect 10C is located as shown by the black box. Unlike the other three previous corrosion defects a faint signature for 10C can be seen.

As Figures K.1, K.2 and K.3 show, three of the four corrosion defects can no longer be observed from data collected from their respective scan lines. In part this is probably due to the passivation of the corrosion products which would have been partly subsumed by the passive layer formed after encasement and also partly due to the increased distance between the probe and the defect as a result of the layer of concrete. This layer was also thicker towards the left side of the mesh which may explain why defects 2C, 3C and 4C are no longer visible, while defect 10C, located on the far right of the mesh can be detected, if only slightly.

Appendix L

Comparison of first and last datasets collected post-concrete encasement

Below are the data collected from each scan line when energising and scanning longitudinally collected during the first week and final week of the mesh experiment for comparison. Defects are labelled based on their visibility in their corresponding scan line and neighbouring scan lines. Defects which were no longer visible after concrete encasement are not labelled.

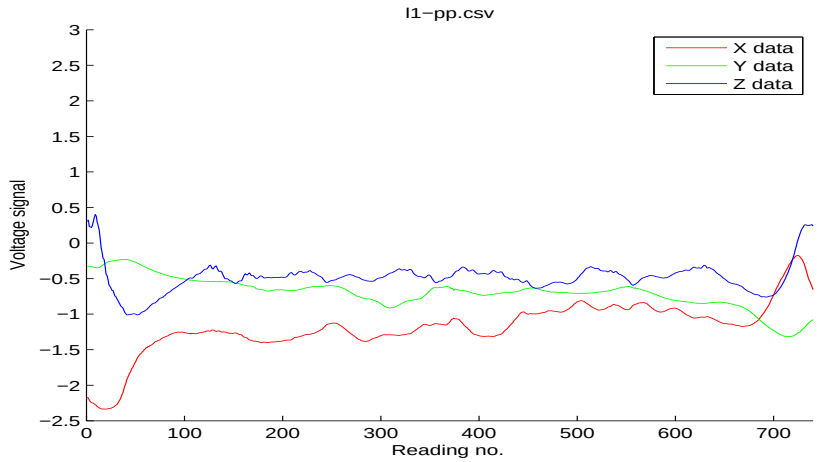


Figure L.1: Data collected from the first week of data collection showing scan line one when energising and scanning longitudinally.

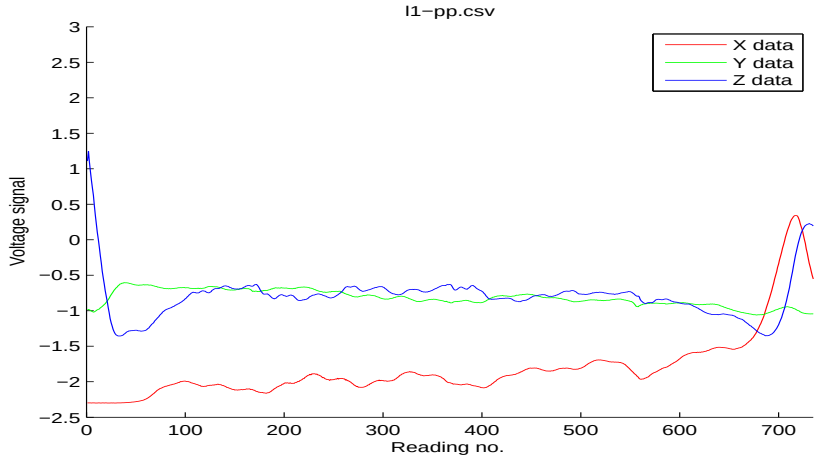


Figure L.2: Data collected from the last week of data collection showing scan line one when energising and scanning longitudinally.

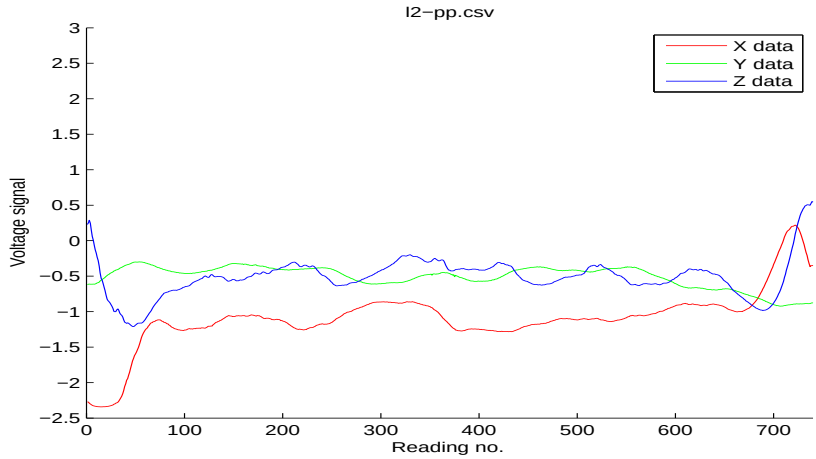


Figure L.3: Data collected from the first week of data collection showing scan line two when energising and scanning longitudinally. Note that defect 1C which should appear around the 667^{th} data point is no longer visible and hence, is not labelled.

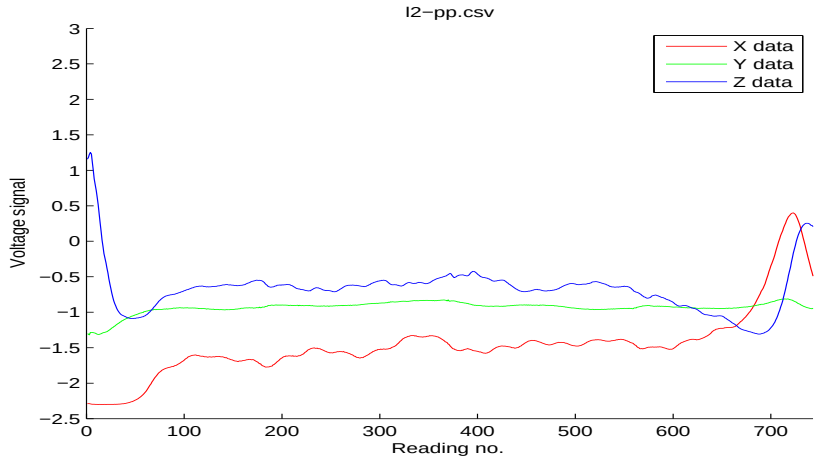


Figure L.4: Data collected from the first week of data collection showing scan line two when energising and scanning longitudinally. Note that defect 1C which should appear around the 667^{th} data point is still no longer visible despite breaking out of the concrete and applying hydrochloric acid to the steel and hence, is not labelled.

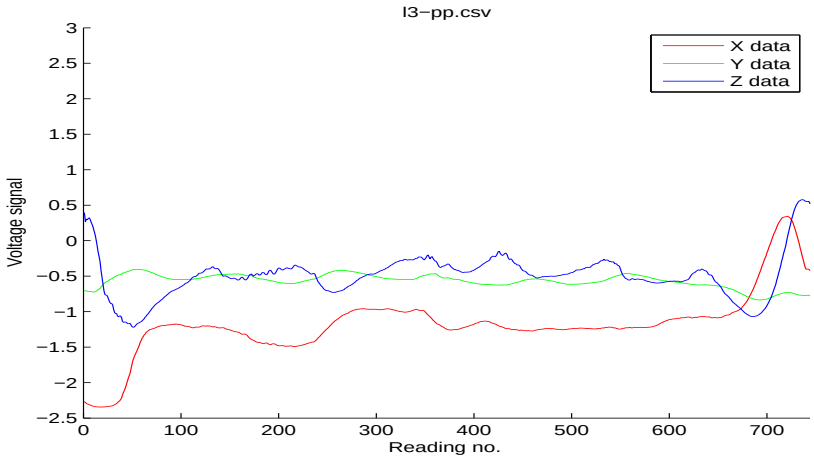


Figure L.5: Data collected from the first week of data collection showing scan line three when energising and scanning longitudinally.

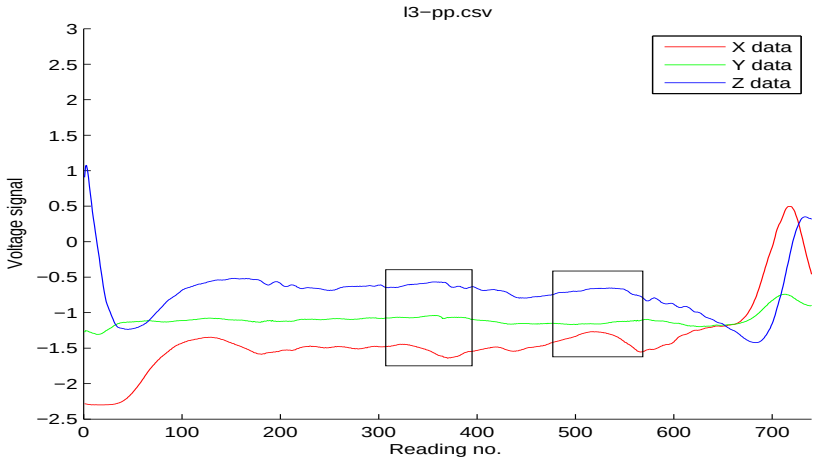


Figure L.6: Data collected from the last week of data collection showing scan line three when energising and scanning longitudinally. Defects 4B and 5B can now be seen from scan line five which could be due to the exacerbation of the damage caused by their breaks during the experiment, or as a result of repeated energisations which may have increased their fringing fields.

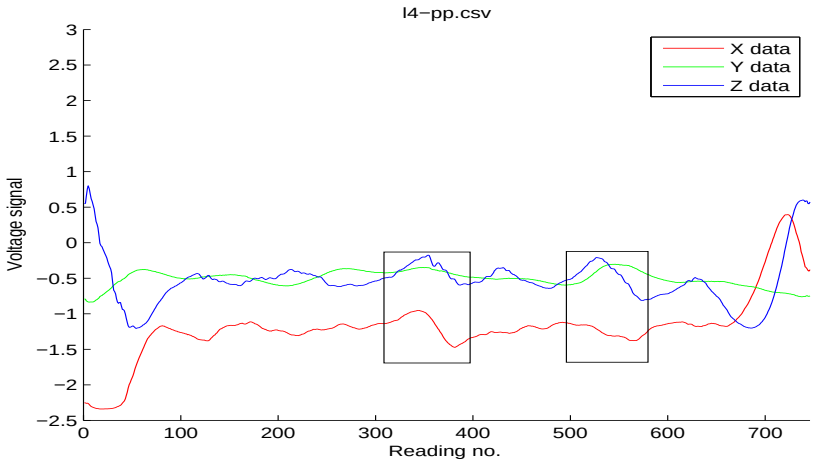


Figure L.7: Data collected from the first week of data collection showing scan line four when energising and scanning longitudinally. Note that defects 2C and 3C are no longer visible after concrete encasement and are not labelled. Defects 4B and 5B from neighbouring scan line five are visible and hence are labelled.

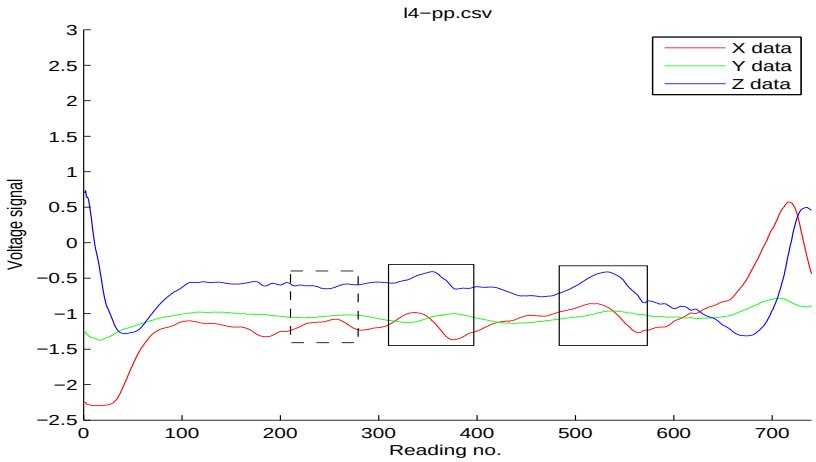


Figure L.8: Data collected from the last week of data collection showing scan line four when energising and scanning longitudinally. A small signature relating to the location of defect 3C is now visible and is labelled (dotted box), as are defects 4B and 5B from neighbouring scan line five. Defect 2C is still difficult to detect in this data, whose signature may be obscured by the end effect towards the start of the scan line.

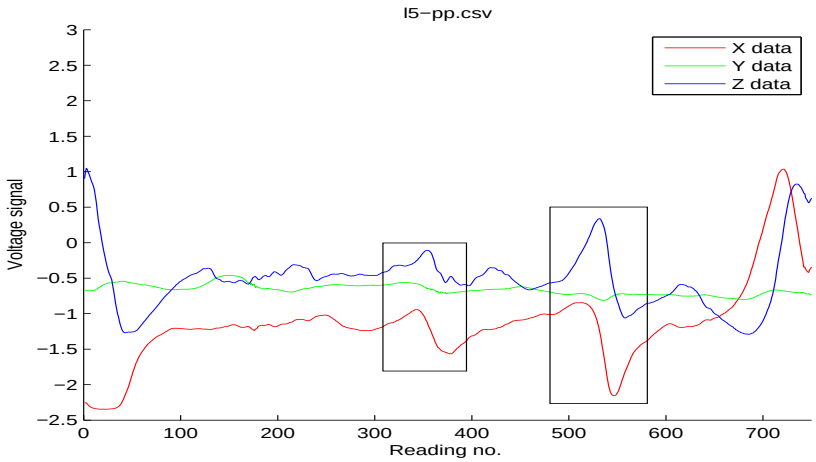


Figure L.9: Data collected from the first week of data collection showing scan line five when energising and scanning longitudinally. Here defects 4B and 5B are visible and are labelled, whereas defect 6BC is not visible when scanning longitudinally, and hence is not labelled.

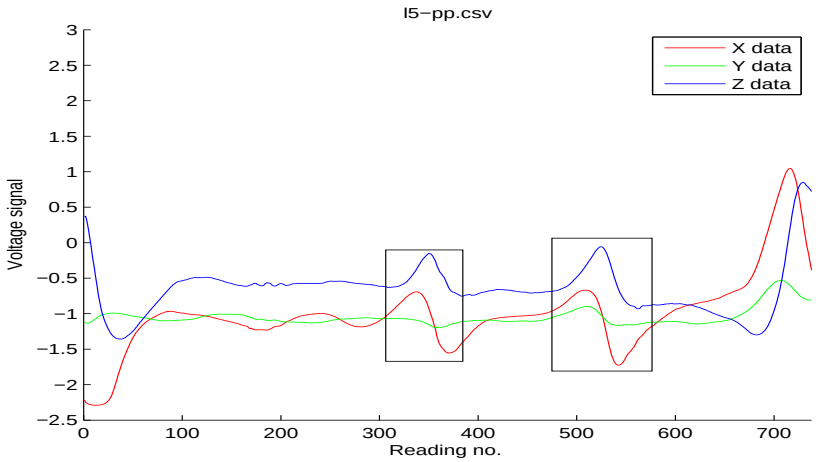


Figure L.10: Data collected from the last week of data collection showing scan line five when energising and scanning longitudinally. Here defects 4B and 5B are still visible and are labelled, whereas defect 6B is still not visible due to the end effect signal and therefore is not labelled. Note that the signatures of defects 4B and 5B have not changed significantly from the first week of data collection, although defect 4B is now clearer along the Z direction.

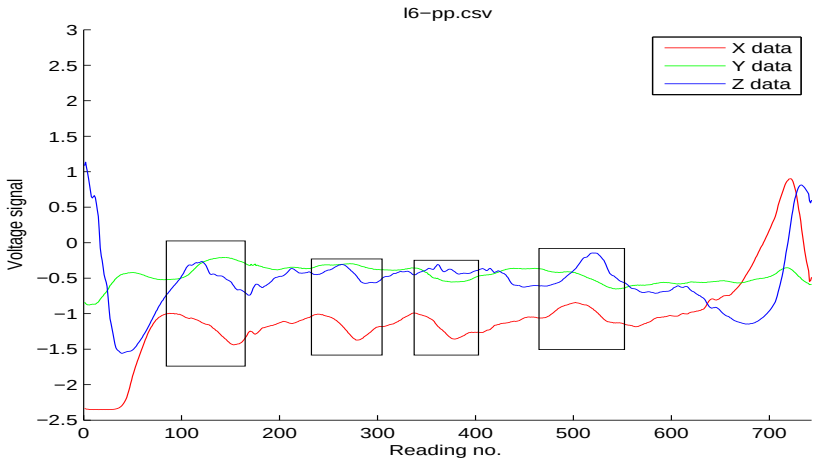


Figure L.11: Data collected from the first week of data collection showing scan line six when energising and scanning longitudinally. Here defects 4B, 5B, 7BC and 8BC are visible and are highlighted. As with scan line five shown in Figure L.9 defect 6BC is still not visible and is not labelled.

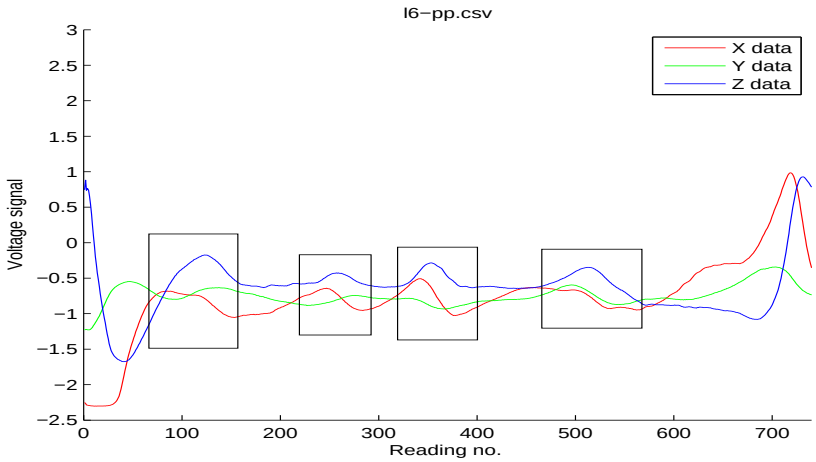


Figure L.12: Data collected from the last week of data collection showing scan line six when energising and scanning longitudinally. Here defects 4B, 5B, 7BC and 8BC are visible and are highlighted. As with scan line five shown in Figure L.10 defect 6BC is still not visible and is not labelled. Here the defect signatures are more defined than the first week's data.

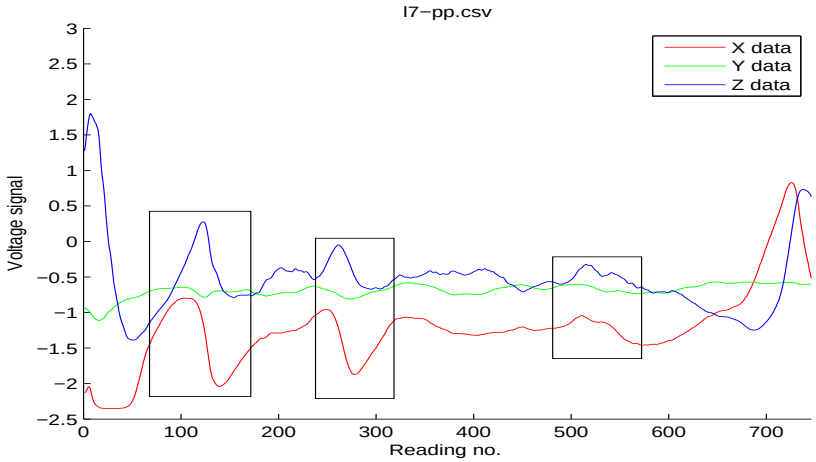


Figure L.13: Data collected from the first week of data collection showing scan line seven when energising and scanning longitudinally. Here defects 7BC and 8BC are visible and are highlighted, as well as a small signature for defect 5B.

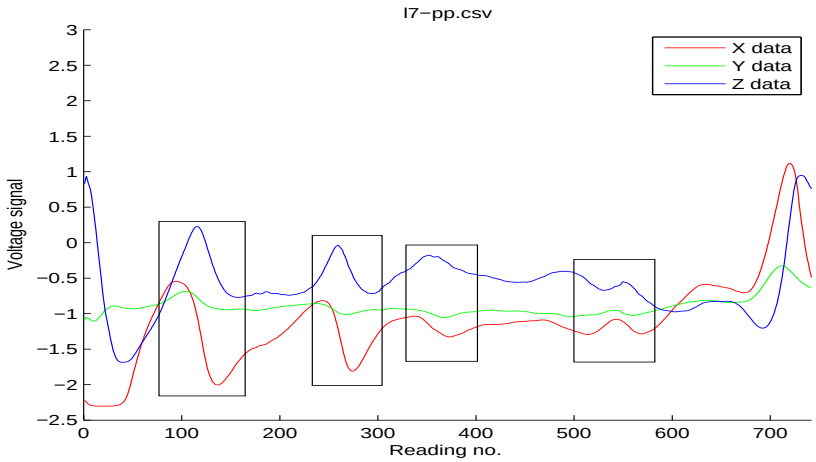


Figure L.14: Data collected from the last week of data collection showing scan line seven when energising and scanning longitudinally. Here defects 7BC and 8BC are visible and are highlighted. In addition to these two defects, defect 4B is now visible and a more defined 5B signature is also highlighted.

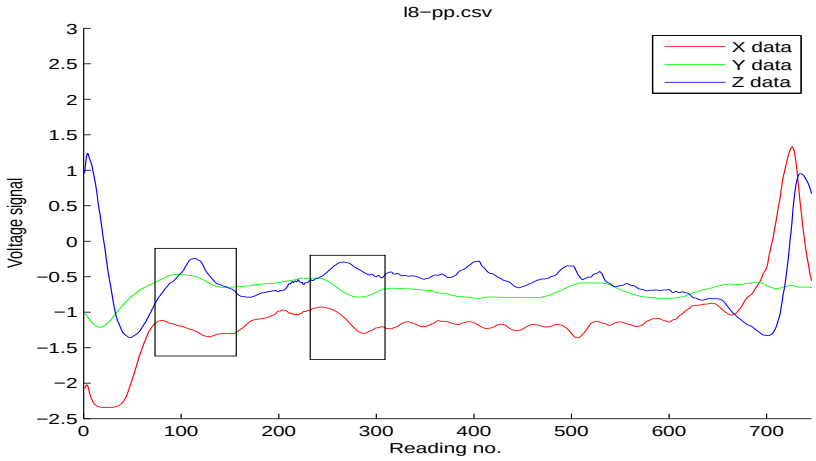


Figure L.15: Data collected from the first week of data collection showing scan line eight when energising and scanning longitudinally. Here defects 7BC and 8BC are visible and are highlighted.

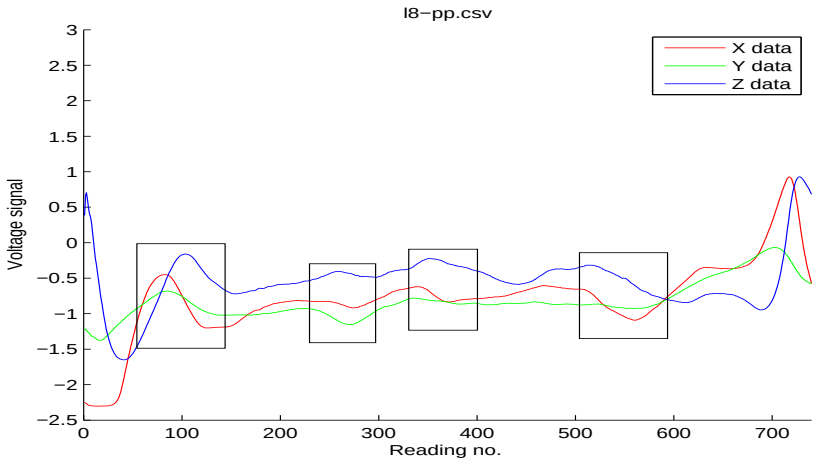


Figure L.16: Data collected from the last week of data collection showing scan line eight when energising and scanning longitudinally. Here defects 7BC, 8BC are visible and are highlighted. In addition signatures from defects 4B and 5B can also be detected and are labelled.

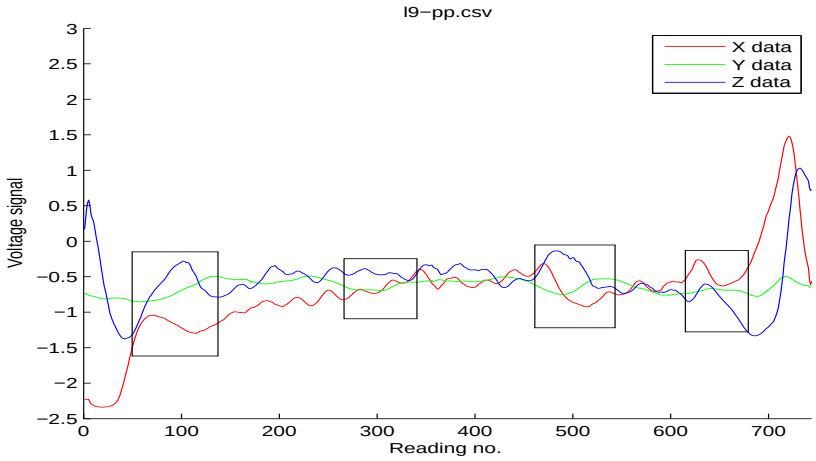


Figure L.17: Data collected from the first week of data collection showing scan line nine when energising and scanning longitudinally. Here defects 9B, 10C, 11BC and 12B are visible and are highlighted.

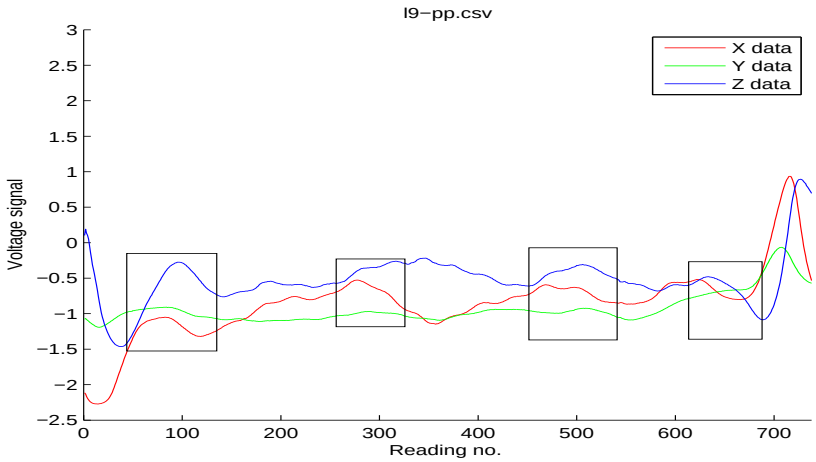


Figure L.18: Data collected from the last week of data collection showing scan line nine when energising and scanning longitudinally. Here defects 9B, 10C, 11BC and 12B are visible and are highlighted. The signatures for all defects apart from 9B appear to have changed when compared to the data collected from the first week of the experiment.

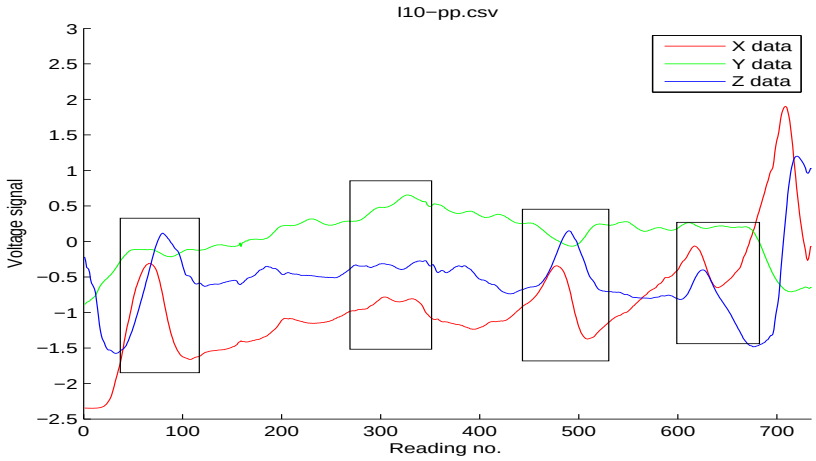


Figure L.19: Data collected from the first week of data collection showing scan line ten when energising and scanning longitudinally. Here defects 9B, 10C, 11BC and 12B are visible and are highlighted.

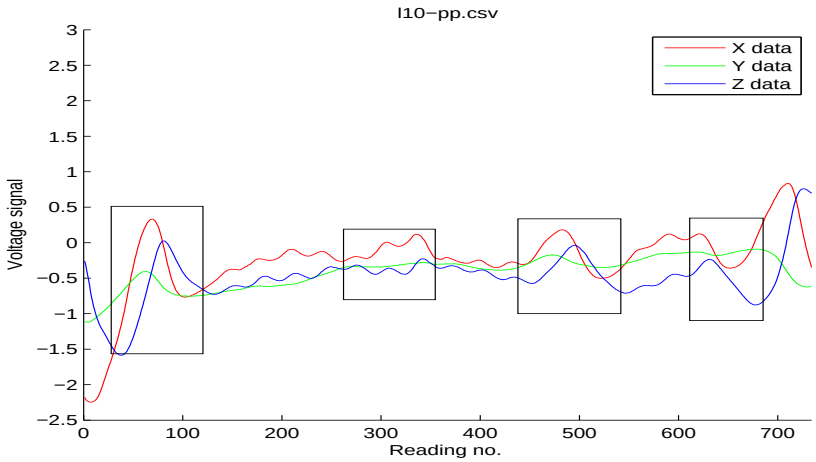


Figure L.20: Data collected from the last week of data collection showing scan line ten when energising and scanning longitudinally. Here defects 9B, 10C, 11BC and 12B are visible and are highlighted. As was the case in Figure L.18, the signature of defect 10 appears to have changed, although the remaining defects are mainly unaltered.

Appendix M

Defect signatures over a 57 week period

Below are plots of the remaining nine scan lines that were not shown in Chapter 5 showing the data collected at five week intervals where the top plot is the 1^{st} week and the bottom plot is the 53^{rd} week. As with all of the other EMAD data shown, the data collected in the X, Y and Z directions are shown by the red, green and blue plots respectively. The Y axis of the graphs show the week number of data collection which is plotted every five weeks. After week 31 shown on the figures, the next week of data collection shown is 38. The data for week 36 should be shown here but is not as a result of an EMAD fault which meant that data could not be collected during weeks 36 and 37 of the data collection duration.

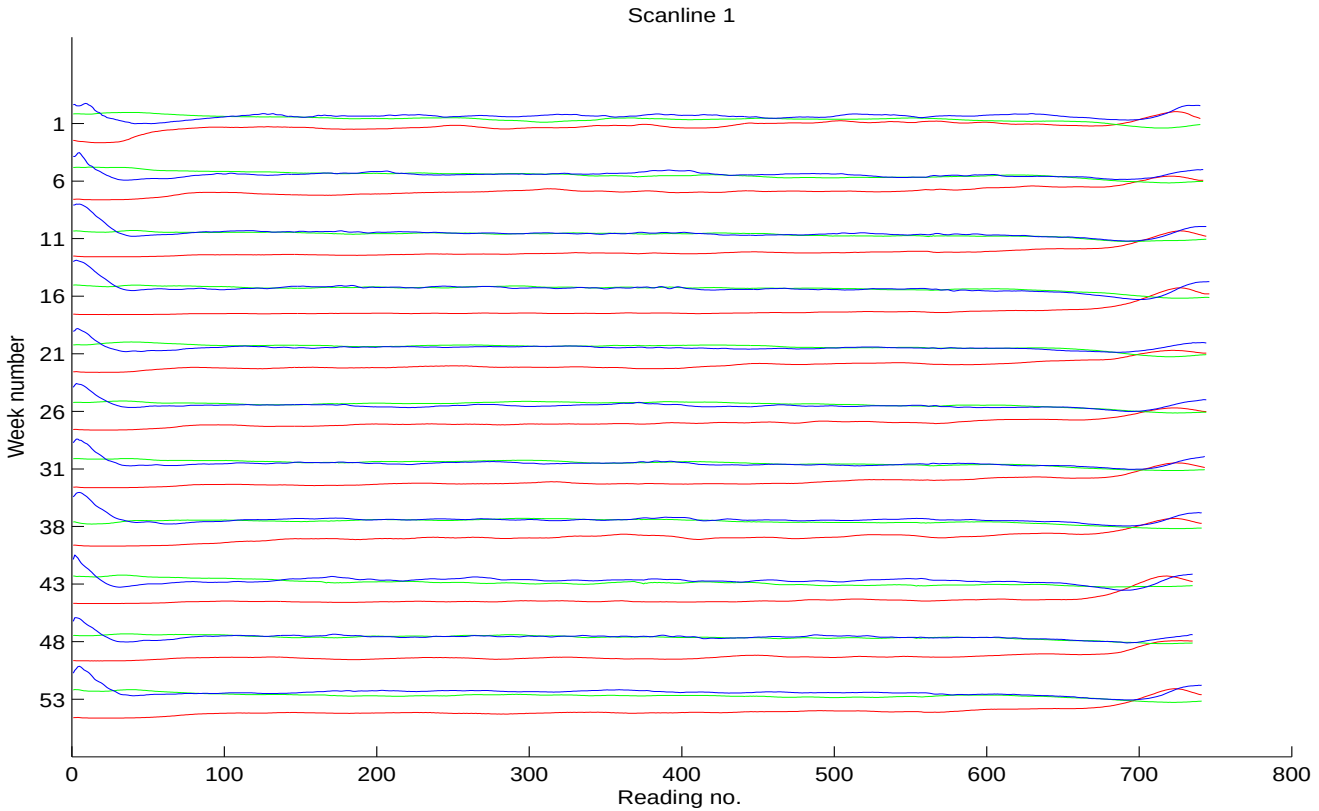


Figure M.1: Data collected during the 57 weeks of data collection from scan line one where each sub plot is data collected from every 5^{th} week.

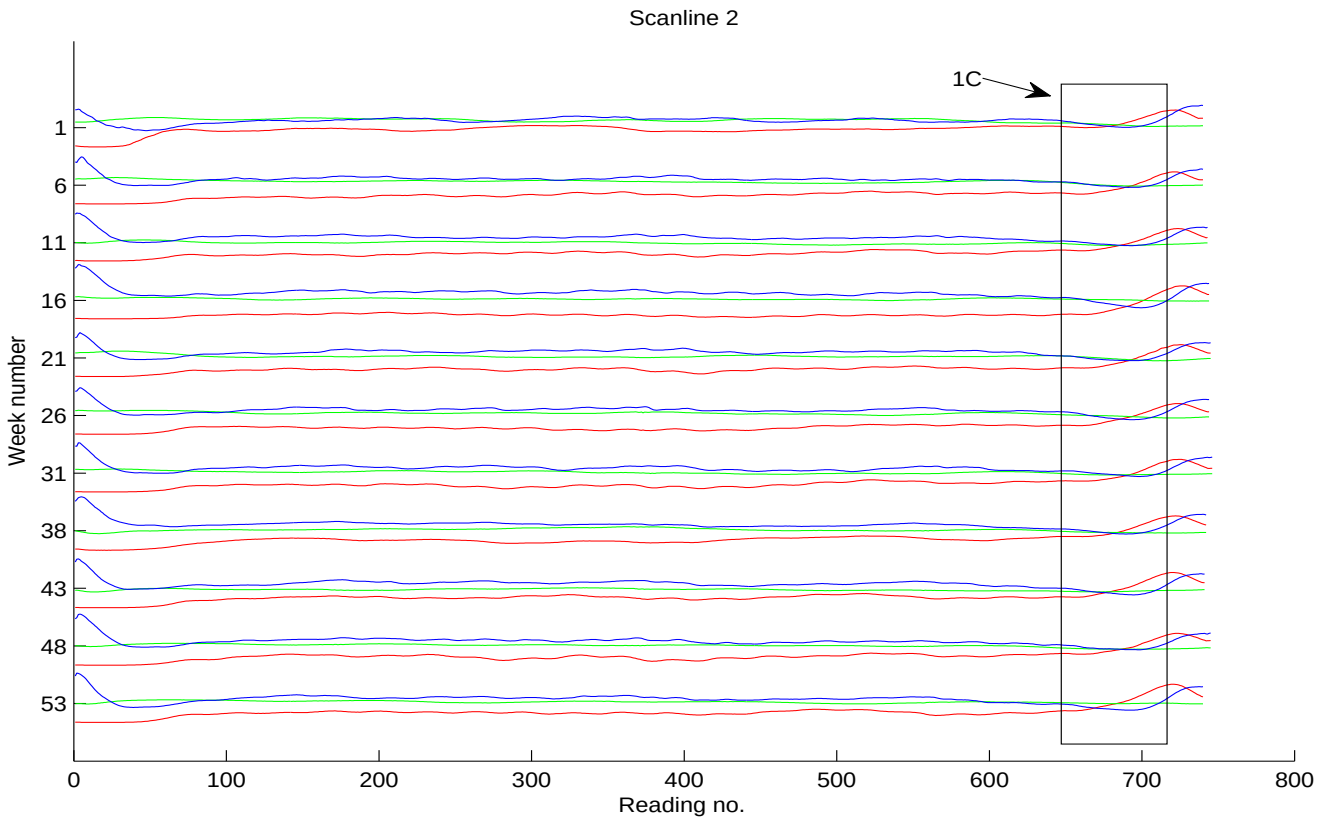


Figure M.2: Data collected during the 57 weeks of data collection from scan line two where each sub plot is data collected from every 5^{th} week. The location of defect 1C is shown where little change is observed.

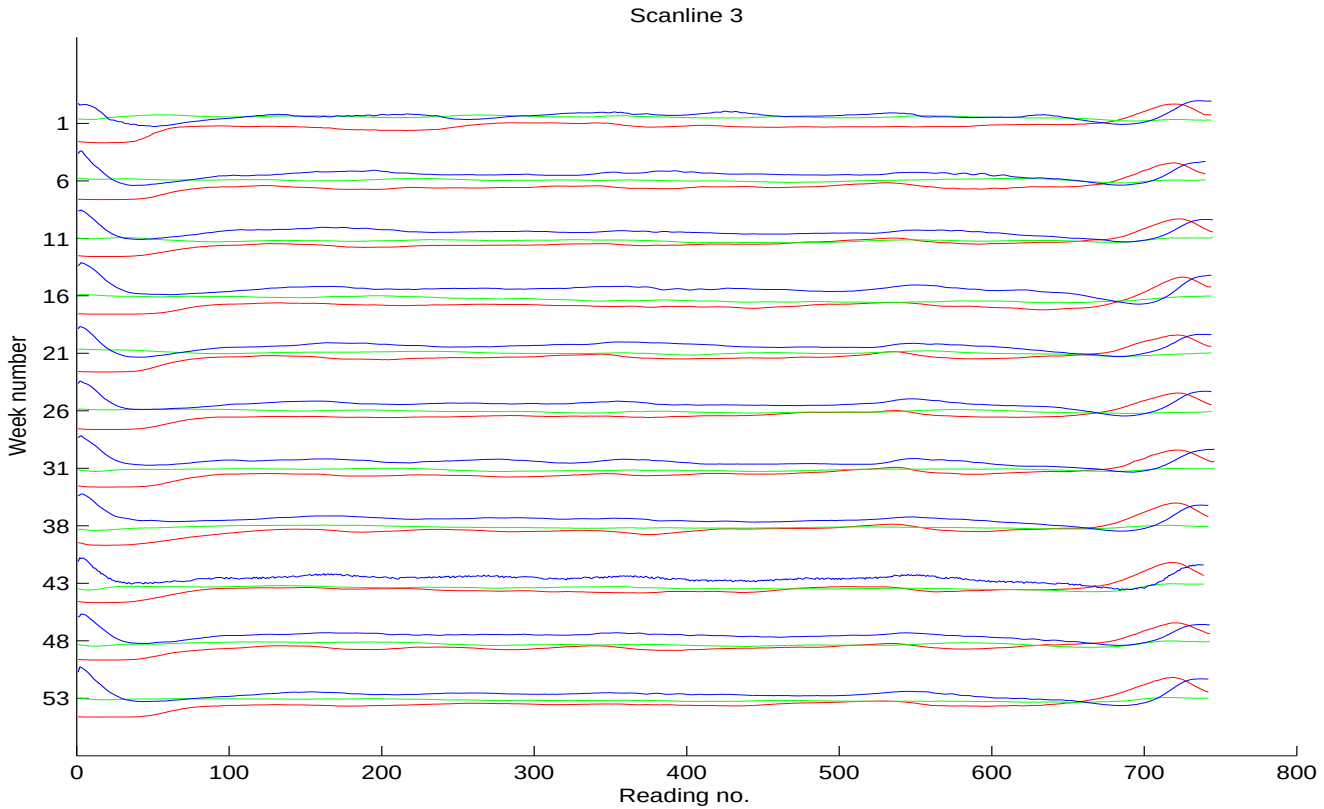


Figure M.3: Data collected during the 57 weeks of data collection from scan line three where each sub plot is data collected from every 5^{th} week.

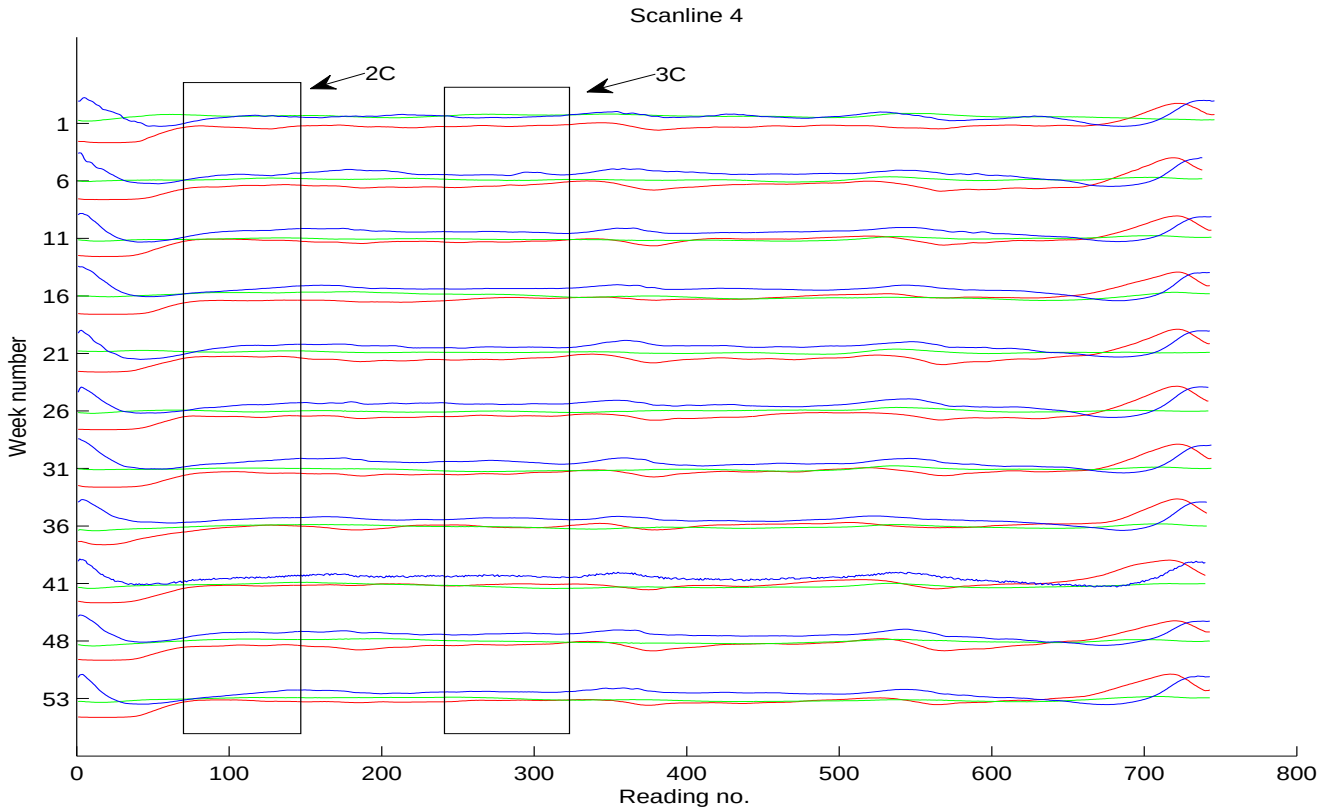


Figure M.4: Data collected during the 57 weeks of data collection from scan line four where each sub plot is data collected from every 5^{th} week. The locations of defects 2C and 3C are shown were little change is observed, while the other defects observable from neighbouring scan lines can be seen but are now labelled.

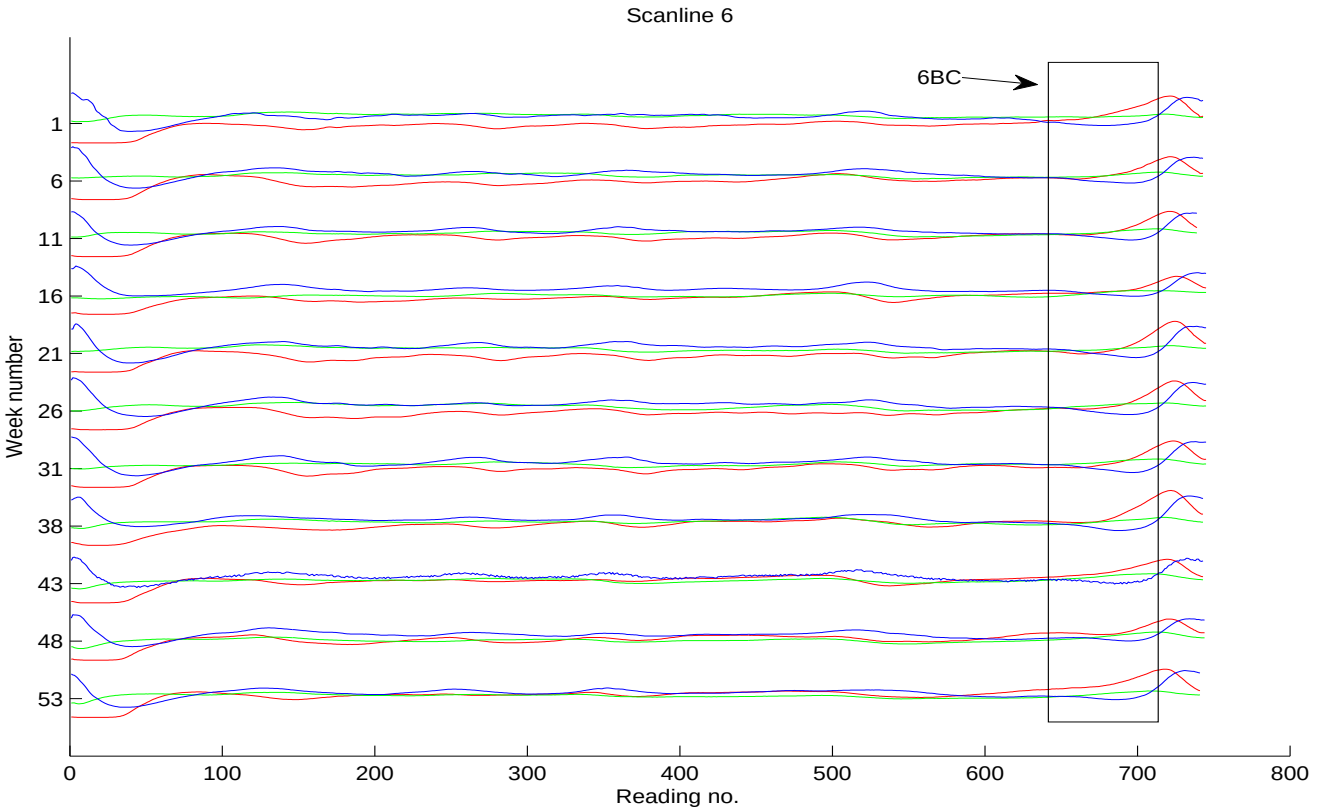


Figure M.5: Data collected during the 57 weeks of data collection from scan line six where each sub plot is data collected from every 5^{th} week. Defects from neighbouring scan lines such as five and seven can be seen but are not labelled for clarity.

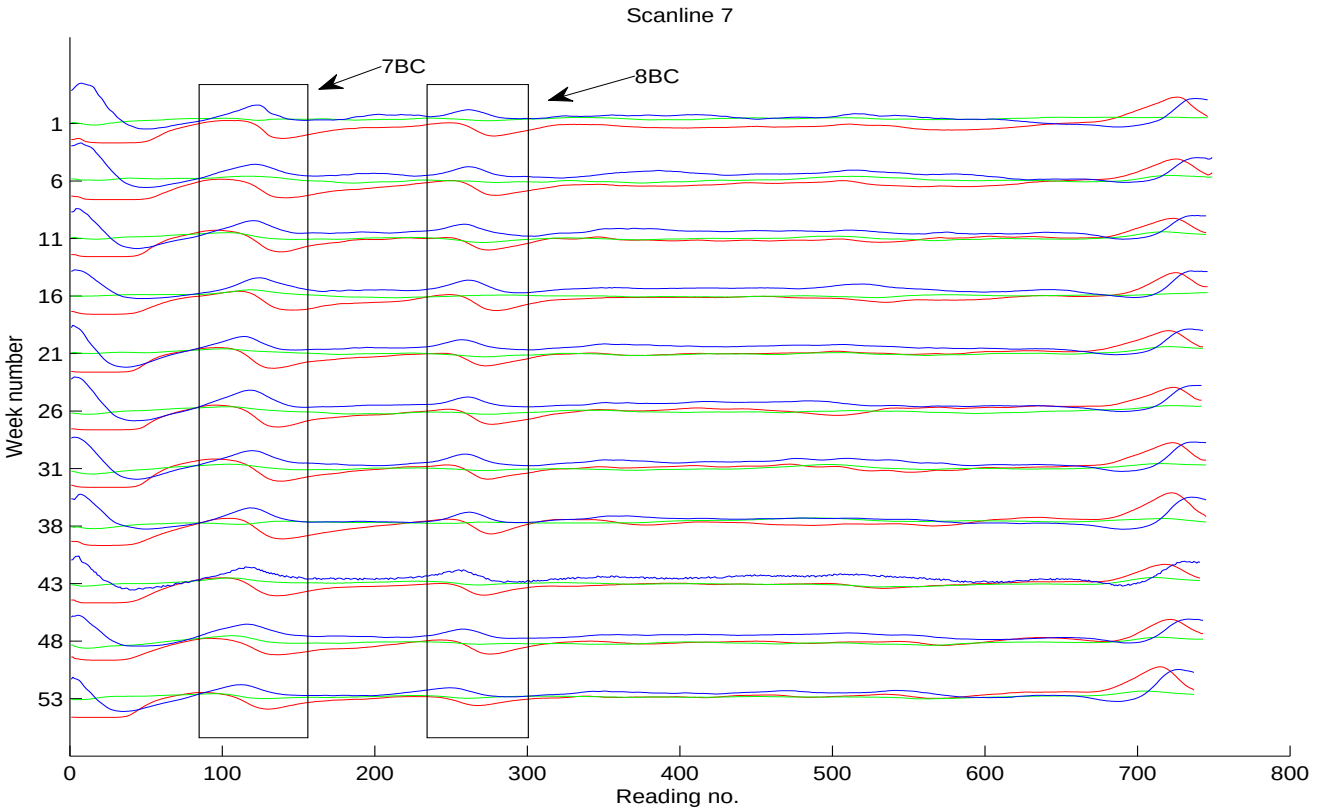


Figure M.6: Data collected during the 57 weeks of data collection from scan line seven where each sub plot is data collected from every 5^{th} week. The location of defects 7BC and 8BC are shown where little change between the first week and week 53 is observable.

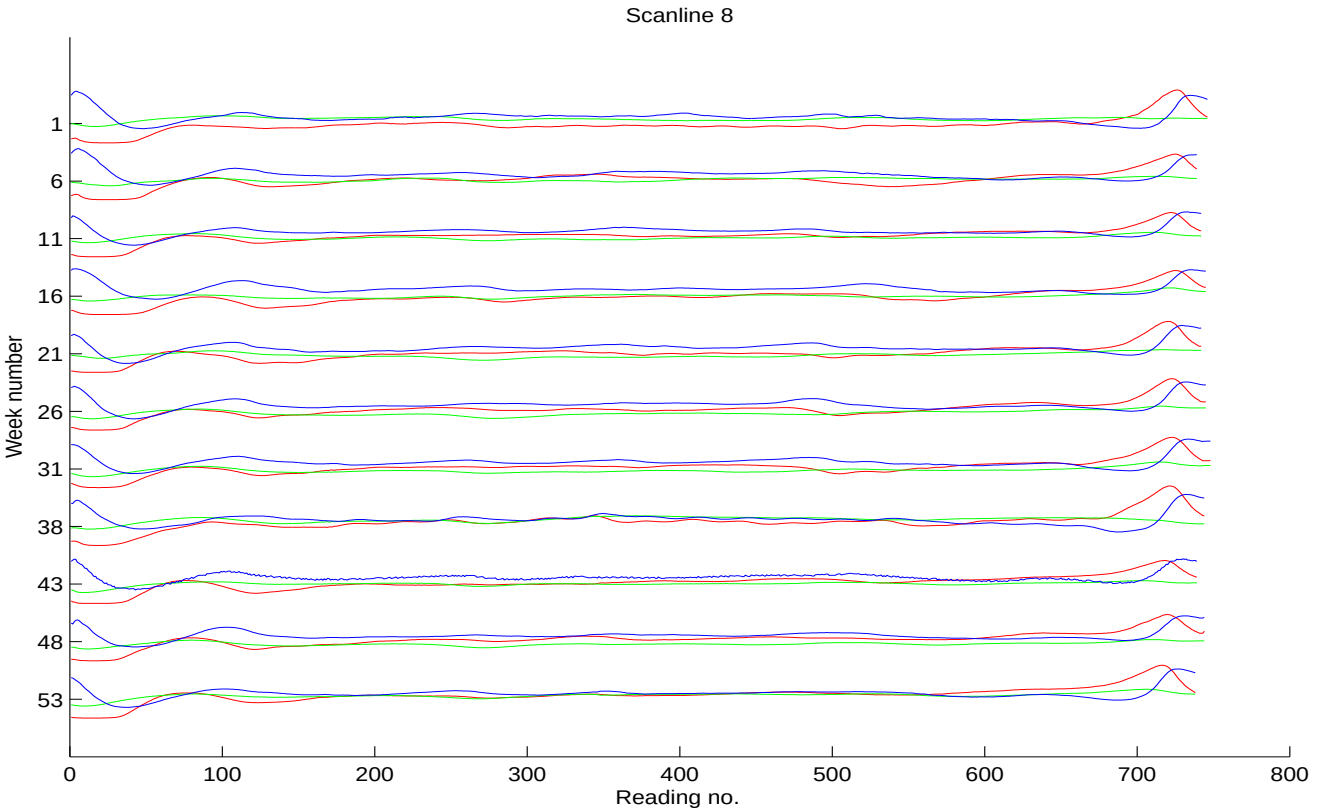


Figure M.7: Data collected during the 57 weeks of data collection from scan line eight where each sub plot is data collected from every 5^{th} week. Defects from neighbouring scan lines such as five and seven can be seen but are not labelled for clarity.

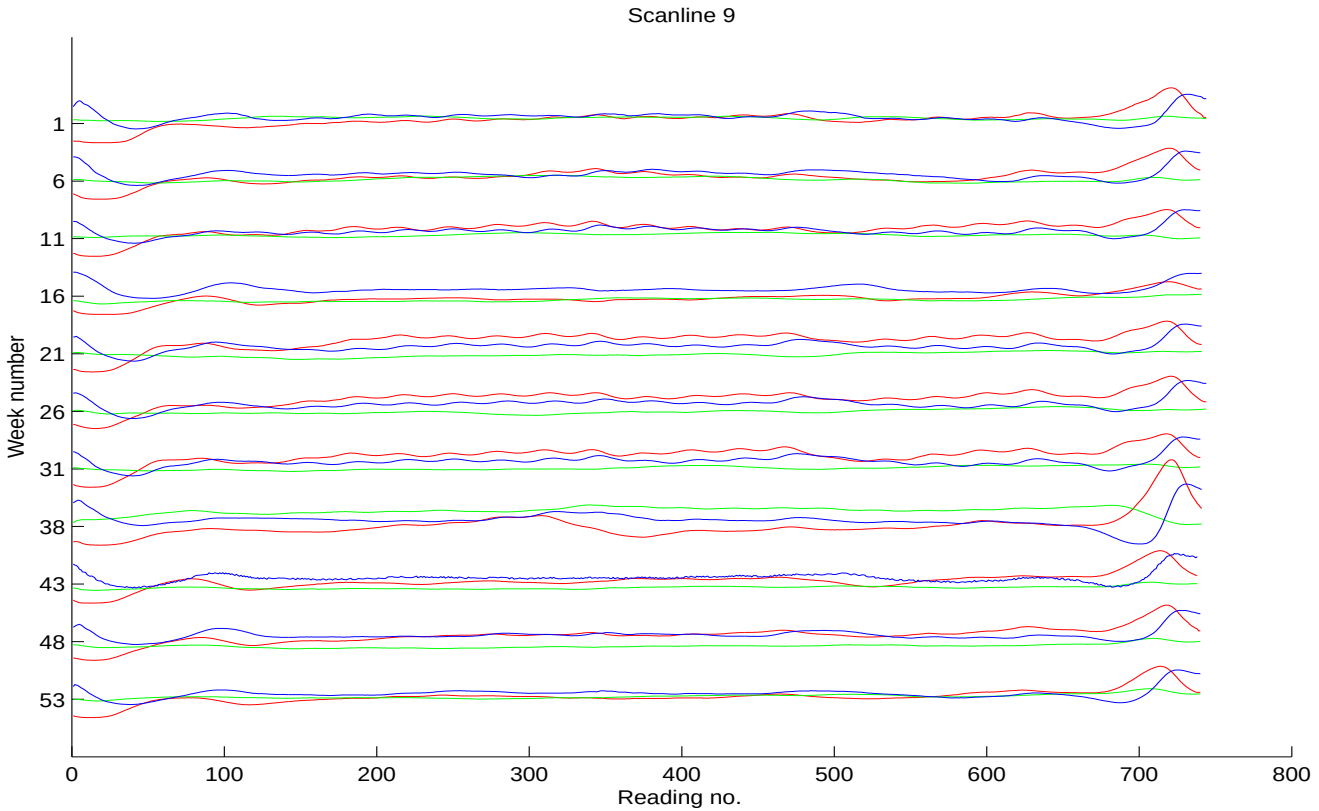


Figure M.8: Data collected during the 57 weeks of data collection from scan line nine where each sub plot is data collected from every 5^{th} week. Defects from neighbouring scan lines such as scan line ten can be seen but are not labelled for clarity.

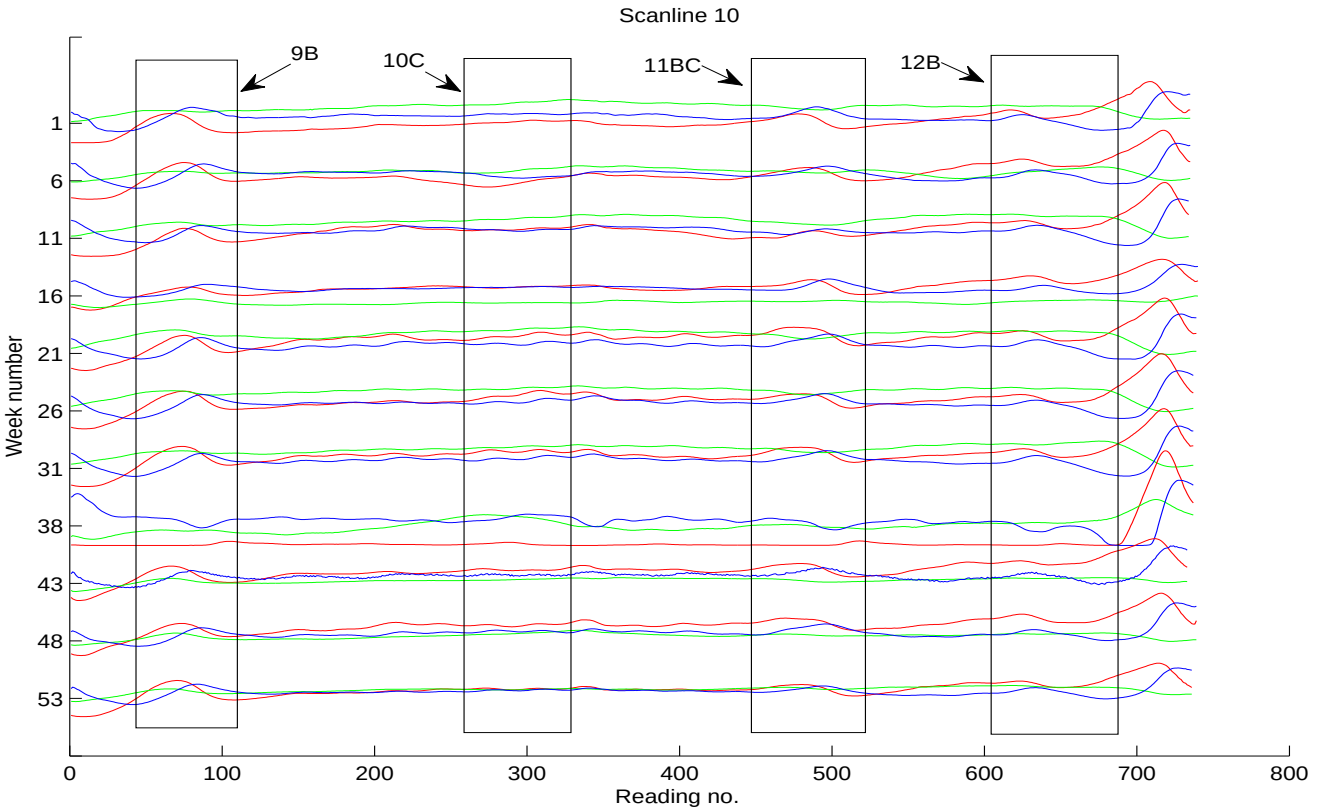


Figure M.9: Data collected during the 57 weeks of data collection from scan line ten where each sub plot is data collected from every 5^{th} week. The location of defects 9B, 10C, 11BC and 12B are shown.